OZON tech

Protobuf в Go

Владислав Сидоров
руководитель отдела
Ozon

AGENDA

01 Введение

02 Типы данных

03 Базовые правила

04 Обновление proto-файлов

05 oneof

06 well-known types

07 Выводы

ozontech

01 Введение

Зачем нам всё это знать?

У пользователя может **не работать требуемый функционал**

У вас возможны **критичные инциденты**

Возможен **избыточный трафик по сети**

Ломаться будет не у вас, а **виноваты будете вы**

Почему в Ozon так важен proto?

Для взаимодействия между собой сервисам НЕОБХОДИМО использовать gRPC

01

gRPC использует proto в качестве базового формата обмена сообщениями

02

Конвенция
по микросервисам

Пример proto-файла

```proto
// Style Guide: имя файла в snake_case.proto

syntax = "proto3";

// Style Guide: имя пакета в нижнем регистре
package ozon.bx.example.api.bar;

option go_package = "gitlab.ozon.ru/bx/example-api/pkg/bar";
option csharp_namespace = "Ozon.Bx.ExampleApi.Bar";
option java_package = "ru.ozon.bx.exampleapi.bar";
option java_multiple_files = true;

// Сущность
message Entity {
  // идентификатор сущности
  uint32 id = 1;
  // алиас сущности
  string alias = 2;
  // название сущности
  string name = 3;
  // признак наличия человеко-понятного URL
  // Style Guide: имя поля пишем в snake_case
  bool is_sef = 5;
}
```

Пример кодогенерации

```go
package bar

//go:generate protoc --proto_path=. --go_out=. --go_opt=paths=source_relative ./entity.proto
```

```go
// Сущность
type Entity struct { 13 usages
    state         protoimpl.MessageState
    sizeCache     protoimpl.SizeCache
    unknownFields protoimpl.UnknownFields

    // идентификатор сущности
    Id uint32 `protobuf:"varint,1,opt,name=id,proto3" json:"id,omitempty"`
    // алиас сущности
    Alias string `protobuf:"bytes,2,opt,name=alias,proto3" json:"alias,omitempty"`
    // название сущности
    Name string `protobuf:"bytes,3,opt,name=name,proto3" json:"name,omitempty"`
    // признак наличия человеко-понятного URL
    // Style Guide: имя поля пишем в snake_case
    IsSef bool `protobuf:"varint,5,opt,name=is_sef,json=isSef,proto3" json:"is_sef,omitempty"`
}
```

02

Типы данных

Скалярные типы

proto	go	json
int32/uint32	int32/uint32	number
int64/uint64	int64/uint64	string
bool	bool	boolean
string	string	string
bytes	[]byte	base64 string
float	float32	number
double	float64	number
в proto нет чисел разрядностью 8 и 16 бит		

Перечисления enum

```proto
// Статус задачи
// Style Guide: наименование Enum пишем в PascalCase
// Style Guide: Значения Enum пишутся в верхнем регистре через прочерк
// Style Guide: CAPITALS_WITH_UNDERSCORES
enum TicketStatus {
  // неопределенный статус
  // Style Guide: нулевое значение должно иметь суффикс `_UNSPECIFIED`
  TICKET_STATUS_UNSPECIFIED = 0;
  // беклог
  BACKLOG =  1;
  // в работе
  WORK_IN_PROGRESS = 2;
  // сделана
  DONE = 3;
  // отменена
  CANCELED= 4;
}
```

Коллекции, repeated

```
12
13    // Упорядоченный список сущно
14    // Style Guide: названия mess
15    message EntityList {
16      // 0 или более элементов
17      // порядок будет сохранен
18      // Style Guide: имя поля до
19      // Entity импортирован из д
20      repeated Entity entities =
21    }
```

```
28    var collectionList = &server.EntityList{  1 usage
29        Entities: []*server.Entity{
30            {
31                Id:    1,
32                Alias: "first",
33                Name:  "Первый",
34            },
35            {
36                Id:    2,
37                Alias: "second",
38                Name:  "Второй",
39                IsSef: true,
40            },
41        },
42    }
```

Коллекции, map

```
23    // Set of entities
24    message EntityMap {
25      // 0 или более элементов
26      // порядок не гарантируется
27      // ключи - любые типы, кроме чис
28      // байтов, перечислений
29      // ключи не могут повторяться
30      // значения - любой message либо
31      // не могут быть значениями непо
32      // Style Guide: имя поля должно
33      // Entity импортирован из другог
34      map<int64, Entity> entities = 1;
35    }
36
37    message MapOfMap {
38      map<int64,EntityMap> my_map = 1;
39    }
40
```

```
11    var collectionMap = &server.EntityMap{  1 usage
12        Entities: map[int64]*server.Entity{
13            1: {
14                Id:    1,
15                Alias: "first",
16                Name:  "Первый",
17            },
18            2: {
19                Id:    2,
20                Alias: "second",
21                Name:  "Второй",
22                IsSef: true,
23            },
24        },
25    }
```

Nested messages

```
41  // Response
42  message Response {
43    EntityList list = 1;
44    string title = 2;
45    // вложенный message
46    // имеет свою нумерацию полей
47    // область видимости не ограничена:
48    // можно использовать, как Response.DetailInfo
49    message DetailInfo {
50      int32 code = 1;
51      string status = 2;
52    }
53    DetailInfo detail_info = 3;
54  }
```

Пример использования

```
      1        a   b   c          n   a    m    e    true
15 [8 1 18 3 97 98 99 26 4 110 97 109 101 40 1]
id:1 alias:"abc" name:"name" is_sef:true
```

```
3   import ...
12
13  func main() {  Vladislav Sidorov *
14      var codec = encoding.GetCodec(proto.Name)
15
16      clientEntity := &client.Entity{
17          Id:    1,
18          Alias: "abc",
19          Name:  "name",
20          IsSef: true,
                                          .Marshal(clientEntity)
                                          торять, установите
                                          _REGISTRATION_CONFLICT=ignore
                                          в настоящих проектах
30
31      serverEntity := &server.Entity{}
32      err = codec.Unmarshal(b, serverEntity)
33      noErr(err)
34
35      fmt.Printf( format: "%d %d\n", len(b), b)
36      fmt.Println(serverEntity.String())
37  }
```

03 Базовые правила

Не полагайтесь на стабильность сериализации данных

```
      1       a  b  c        n  a   m   e   true
15 [8 1 18 3 97 98 99 26 4 110 97 109 101 40 1]
id:1 alias:"abc" name:"name" is_sef:true
```

```
31      b = []byte{
32          26, 4, 110, 97, 109, 101, // field 3, value "name"
33          8, 1, // field 1, value 1
34          40, 1, // field 4, value bool
35          18, 3, 97, 98, 99, // field 2, value "abc"
36      }
37      serverEntity := &server.Entity{}
38      err = codec.Unmarshal(b, serverEntity)
39      noErr(err)
40
41      fmt.Printf( format: "%d %d\n", len(b), b)
42      fmt.Println(serverEntity.String())
43  }
```

```
          n  a   m   e   1   true     a  b  c
15 [26 4 110 97 109 101 8 1 40 1 18 3 97 98 99]
id:1 alias:"abc" name:"name" is_sef:true
```

Реальный пример

```
125     result := &componentpb.ComponentState{
126         ComponentId: instance.ComponentId,
127     }
128
129     switch instance.View.String() {
130     case taglistV1.ViewV1.String():
131         // case tag listV1
132         // ...
133         result.Data = data
134     case taglistV2.View.String():
135         // case tag listV2
136         // ...
137         result.Data = data
138     }
139
140     return result, nil, nil
141 }
```

Часто используемые сообщения выносите в отдельные файлы

```proto
syntax = "proto3";

import "entity.proto";

package ozon.bx.example.api.bar;

option go_package = "gitlab.ozon.ru/bx/example-api/pkg/bar";
option csharp_namespace = "Ozon.Bx.ExampleApi.Bar";
option java_package = "ru.ozon.bx.exampleapi.bar";
option java_multiple_files = true;


// Упорядоченный список сущностей
// Style Guide: названия message пишем в PascalCase
message EntityList {
  // 0 или более элементов
  // порядок будет сохранен
  // Style Guide: имя поля должно быть множественным
  // Entity импортирован из другого proto
  repeated Entity entities = 1;
}
```

Поля, заполненные значениями, по умолчанию не сериализуются

```go
13  func main() {  Vladislav Sidorov *
14      var codec = encoding.GetCodec(proto.Name)
15
16      clientEntity := &client.Entity{
17          Id:    1,
18          Alias: "abc",
19          Name:  "",
20          IsSef: true,
21      }
22
23      b, err := codec.Marshal(clientEntity)
24      noErr(err)
25      /*...*/
30
31      serverEntity := &server.Entity{}
32      err = codec.Unmarshal(b, serverEntity)
33      noErr(err)
34
35      fmt.Printf( format: "%d %d\n", len(b), b)
36      fmt.Println(serverEntity.String())
37  }
```

```
      1       a  b  c   true
9 [8 1 18 3 97 98 99 40 1]
id:1 alias:"abc" is_sef:true
```

Номера полей до 15 включительно занимают 1 байт

```
13  // Номера полей
14  message Nums {
15    // номера полей 1-15 занимают 1 байт
16    uint32 id_1 = 1;
17    uint32 id_15 = 15;
18    // номера полей 16-2047 занимают 2 байта
19    uint32 id_16 = 16;
20    uint32 id_2047 = 2047;
21    // номера полей 2048-262143 занимают 3 байта
22    uint32 id_2048 = 2048;
23    // номера полей 19000-19999
24    // зарезервированы для protobuf
25    uint32 id_262143 = 262143;
26    // 4 байта
27    uint32 id_262144 = 262144;
28    uint32 id_33554431 = 33554431;
29    // 5 байт
30    uint32 id_33554432 = 33554432;
31    uint32 id_536870911 = 536870911;
32  }
```

```
10      func numFields() {  1 usage
11          var codec = encoding.GetCodec(proto.Name)
12
13          nums := &server.Nums{
14              Id_1:    100,
15              Id_15:   101,
16              Id_16:   102,
17              Id_2047: 103,
18              Id_2048: 104,
19          }
20
21          b, err := codec.Marshal(nums)
22          noErr(err)
23          fmt.Printf( format: "%d %d\n%s\n",
24              len(b), b, nums)
25      }
```

```
      100      101       102          103            104
14 [8 100 120 101 128 1 102 248 127 103 128 128 1 104]
id_1:100 id_16:102 id_15:101 id_2047:103 id_2048:104
```

Пишите комментарии

Это очевидно.
Пользоваться будете не только вы,
а в основном — ваши клиенты

01

Есть поддержка однострочных
`//` и `/* ... */` многострочных комментариев

02

04

Обновление proto-файлов

Обновление proto-файлов

- Изначально у сервера и клиентов одинаковые proto-файлы
- По мере развития сервера появляются новые потребности
- В результате у сервера и клиентов оказываются разные proto-файлы

ВАЖНО

Никогда **нельзя** менять номера существующих полей

Не делайте так

```
13  // Сущность
14  message Entity {
15    // идентификатор сущности
16    uint32 id = 1;
17    // алиас сущности
18    string alias = 2;
19    // название сущности
20    string name = 3;
21    // признак наличия человеко-понятного URL
23        bool is_sef = 4;
24  }
```

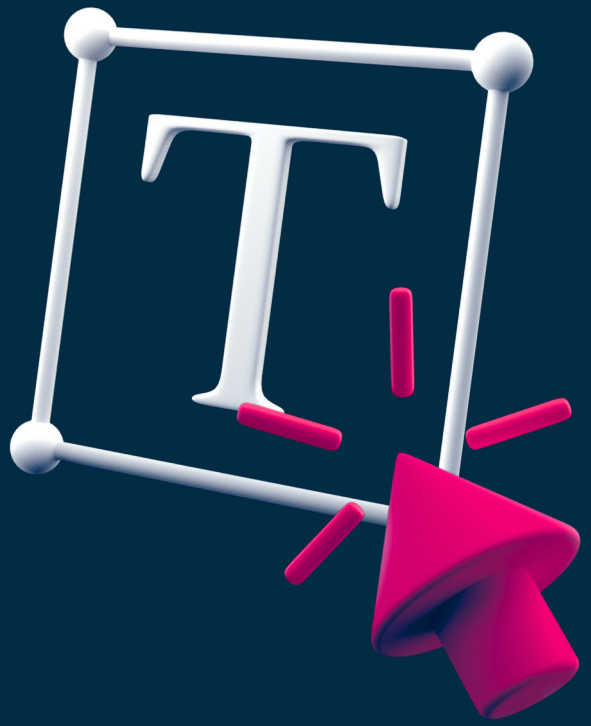
```
serverEntity = {*bar.Entity | 0xc00006e360}
  state = {impl.MessageState}
  sizeCache = {int32} 0
  unknownFields = {[]uint8} len:2, cap:8 ... View
  Id = {uint32} 1
  Alias = {string} "abc"
  Name = {string} "name"
  IsSef = {bool} false
```

```
13  func main() {   Vladislav Sidorov *
14      var codec = encoding.GetCodec(proto.Name)
15
16      clientEntity := &client.Entity{
17          Id:    1,
18          Alias: "abc",
19          Name:  "name",
20          IsSef: true,
21      }
22
23      b, err := codec.Marshal(clientEntity)
24      noErr(err)
25      /*...*/
30
31      serverEntity := &server.Entity{}
32      err = codec.Unmarshal(b, serverEntity)
33      noErr(err)
34
35      fmt.Printf( format: "%d %d\n", len(b), b)
36      fmt.Println(serverEntity.String())
37  }
```

```
      1       a  b  c        n  a   m   e   true
15 [8 1 18 3 97 98 99 26 4 110 97 109 101 40 1]
id:1 alias:"abc" name:"name" 5:1
```

С точки зрения proto поля и имена полей можно переименовывать

Переименование полей



- Это безопасно с точки зрения proto-совместимости
- Если на базе proto есть генерация json, то это ломает контракт
- Также переименование полей может вызвать неудобство актуализации имён полей у клиентов

Переименование полей

```
13      // Сущность
14      message Entity {
15        // идентификатор сущности
16        uint32 sku = 1;
18        string alias = 2;
19        // название сущности
20        string name = 3;
21        // признак наличия человеко-понятного URL
22        // Style Guide: имя поля пишем в snake_case
23        bool is_sef = 5;
24      }
```

```
      1       a  b  c        n  a   m   e   true
15 [8 1 18 3 97 98 99 26 4 110 97 109 101 40 1]
sku:1  alias:"abc"  name:"name"  is_sef:true
1 name
```

```
13  func main() {  Vladislav Sidorov *
14      var codec = encoding.GetCodec(proto.Name)
15      clientEntity := &client.Entity{
16          Id:    1,
17          Alias: "abc",
18          Name:  "name",
19          IsSef: true,
20      }
21
22      b, err := codec.Marshal(clientEntity)
23      noErr(err)
24      /*...*/
29
30      serverEntity := &server.Entity{}
31      err = codec.Unmarshal(b, serverEntity)
32      noErr(err)
33
34      fmt.Printf( format: "%d %d\n", len(b), b)
35      fmt.Println(serverEntity.String())
36      fmt.Println(serverEntity.Sku, serverEntity.Name)
37  }
```

В большинстве случаев нельзя менять тип полей

Не делайте так

```proto
// Сущность
message Entity {
  // идентификатор сущности
  string id = 1;
  string alias = 2;
  // название сущности
  string name = 3;
  // признак наличия человеко-понятного URL
  // Style Guide: имя поля пишем в snake_case
  bool is_sef = 5;
}
```

```
serverEntity = {*bar.Entity | 0xc00006e360}
  state = {impl.MessageState}
  sizeCache = {int32} 0
  unknownFields = {[]uint8} len:2, cap:8 ... View
  Id = {string} ""
  Alias = {string} "abc"
  Name = {string} "name"
  IsSef = {bool} true
```

```go
func main() {  Vladislav Sidorov *
    var codec = encoding.GetCodec(proto.Name)

    clientEntity := &client.Entity{
        Id:    1,
        Alias: "abc",
        Name:  "name",
        IsSef: true,
    }

    b, err := codec.Marshal(clientEntity)
    noErr(err)
    /*...*/

    serverEntity := &server.Entity{}
    err = codec.Unmarshal(b, serverEntity)
    noErr(err)

    fmt.Printf( format: "%d %d\n", len(b), b)
    fmt.Println(serverEntity.String())
}
```

```
      1       a   b   c         n   a    m    e    true
15 [8 1 18 3 97 98 99 26 4 110 97 109 101 40 1]
alias:"abc"  name:"name"  is_sef:true  1:1
```

С точки зрения proto поля удалять можно

Удаление полей

- Это не ломает контракт proto
- Но лучше перед удалением добавить префикс **OBSOLETE_** , тег **[deprecated = true]** и комментарий
- Резервируйте номера удаляемых полей, чтобы их никто не переиспользовал

Удаление полей

```proto
// Сущность
message Entity {
  // идентификатор сущности
  uint32 id = 1;
  // алиас сущности
  string alias = 2;

  // название сущности
  // не используйте его, оно будет удалено
  string OBSOLETE_name = 3 [deprecated = true];

  bool is_sef = 5;
}
```

```
      1       a  b  c        n  a   m   e   true
15 [8 1 18 3 97 98 99 26 4 110 97 109 101 40 1]
id:1 alias:"abc" OBSOLETE_name:"name" is_sef:true
1 name
```

```go
func main() {  Vladislav Sidorov *
    var codec = encoding.GetCodec(proto.Name)
    clientEntity := &client.Entity{
        Id:    1,
        Alias: "abc",
        Name:  "name",
        IsSef: true,
    }

    b, err := codec.Marshal(clientEntity)
    noErr(err)
    /*...*/

    serverEntity := &server.Entity{}
    err = codec.Unmarshal(b, serverEntity)
    noErr(err)

    fmt.Printf( format: "%d %d\n", len(b), b)
    fmt.Println(serverEntity.String())
    fmt.Println(serverEntity.Id, serverEntity.OBSOLETEName)
}

func noEr
    if er
        f
        o
    }
```

'OBSOLETEName' is deprecated

field OBSOLETEName string`protobuf:"bytes,3,opt,name=O

название сущности не используйте его, оно будет удалено
Deprecated: Do not use.
`OBSOLETEName` on pkg.go.dev ↗

Резервирование полей

```
13    // Сущность
14    message Entity {
15      // идентификатор сущности
16      uint32 id = 1;
17      // алиас сущности
18      string alias = 2;
19      reserved 3;
22      // признак наличия человеко-понятного URL
23      // Style Guide: имя поля пишем в snake_case
24      bool is_sef = 5;
25    }
```

```
      1       a  b  c        n  a   m   e   true
15 [8 1 18 3 97 98 99 26 4 110 97 109 101 40 1]
id:1  alias:"abc"  is_sef:true  3:"name"
```

```
13  func main() {   Vladislav Sidorov *
14      var codec = encoding.GetCodec(proto.Name)
15      clientEntity := &client.Entity{
16          Id:            1,
17          Alias:         "abc",
18          OBSOLETEName: "name",
19          IsSef:         true,
20      }
21
22      b, err := codec.Marshal(clientEntity)
23      noErr(err)
24      /*...*/
29
30      serverEntity := &server.Entity{}
31      err = codec.Unmarshal(b, serverEntity)
32      noErr(err)
33
34      fmt.Printf( format: "%d %d\n", len(b), b)
35      fmt.Println(serverEntity.String())
36  }
```

Можно добавлять новые поля

05 { oneof

oneof

```
10  message Metadata {
11    string key = 1;
12    // value будет содержать не более одного поля
13    oneof value {
14      int32 int_value = 2;
15      string string_value = 3;
16    }
17  }
```

```
28  type Metadata struct {  14 usages
29      state         protoimpl.MessageState
30      sizeCache     protoimpl.SizeCache
31      unknownFields protoimpl.UnknownFields
32
33      Key string `protobuf:"bytes,1,opt,name=key,proto3" json:"key,omitempty"`
34      // value будет содержать не более одного поля
35      //
36      // Types that are assignable to Value:
37      //  *Metadata_IntValue
38      //  *Metadata_StringValue
39      Value isMetadata_Value `protobuf_oneof:"value"`
40  }
```

```
102  type isMetadata_Value interface {  2 usages  2 implemen
103      isMetadata_Value()  2 implementations
104  }
105
106  type Metadata_IntValue struct {  5 usages
107      IntValue int32 `protobuf:"varint,2,opt,name=int_v
108  }
109
110  type Metadata_StringValue struct {  5 usages
111      StringValue string `protobuf:"bytes,3,opt,name=st
112  }
113
114  func (*Metadata_IntValue) isMetadata_Value() {}  no us
115
116  func (*Metadata_StringValue) isMetadata_Value() {}  n
```

```
5   var metadataString = &server.Metadata{  1 us
6       Key: "size",
7       Value: &server.Metadata_StringValue{
8           StringValue: "L",
9       },
10  }
11
12  var metadataInt = &server.Metadata{  1 usage
13      Key: "size",
14      Value: &server.Metadata_IntValue{
15          IntValue: 48,
16      },
17  }
```

06 well-known types

well known types

```proto
syntax = "proto3";

import "google/protobuf/wrappers.proto";
import "google/protobuf/duration.proto";

package ozon.bx.example.api.bar;

option go_package = "gitlab.ozon.ru/bx/example-api/pkg/bar";
option csharp_namespace = "Ozon.Bx.ExampleApi.Bar";
option java_package = "ru.ozon.bx.exampleapi.bar";
option java_multiple_files = true;

// Имя человека
message PersonName {
  // собственно имя
  string first_name = 1;
  // фамилия
  string last_name = 2;
  // отчество (если есть)
  google.protobuf.StringValue second_name = 3;
}

message RaceResults {
  PersonName person = 1;
  int32 number = 2;
  google.protobuf.Duration result = 3;
}
```

```go
var personOne = &server.PersonName{  1 usage    Vladislav Sidorov
    FirstName: "Иванов",
    LastName:  "Иван",
    SecondName: &wrappers.StringValue{
        Value: "Иванович",
    },
}

var personTwo = &server.PersonName{  1 usage    Vladislav Sidorov
    FirstName: "Бьёрн",
    LastName:  "Страуструп",
}

var raceResults = &server.RaceResults{  1 usage    Vladislav Sido
    Person: personOne,
    Number: 2,
    Result: &durationpb.Duration{
        Seconds: 26,
        Nanos:   400_000_000,
    },
}

var raceDuration = raceResults.Result.AsDuration()  1 usage
```

07

Выводы

Выводы

Никогда не меняйте номера существующих полей

01

Выводы

Никогда не меняйте номера существующих полей

01

Резервируйте номера удаляемых полей

02

Выводы

Никогда не меняйте номера существующих полей **01**

Резервируйте номера удаляемых полей **02**

Protobuf не страшен, если вы не меняете номера существующих полей **03**

Выводы

Никогда не меняйте номера существующих полей **01**

Резервируйте номера удаляемых полей **02**

Protobuf не страшен, если вы не меняете номера существующих полей **03**

Не следует менять типы существующих полей **04**

Выводы

Никогда не меняйте номера существующих полей **01**

Резервируйте номера удаляемых полей **02**

Protobuf не страшен, если вы не меняете номера существующих полей **03**

Не следует менять типы существующих полей **04**

Частые поля должны иметь номер поля до 15 включительно **05**

Выводы

Никогда не меняйте номера существующих полей 01

Резервируйте номера удаляемых полей 02

Protobuf не страшен, если вы не меняете номера существующих полей 03

Не следует менять типы существующих полей 04

Частые поля должны иметь номер поля до 15 включительно 05

Не рассчитывайте на стабильность бинарных и текстовых данных 06

Выводы

Никогда не меняйте номера существующих полей 01

Резервируйте номера удаляемых полей 02

Protobuf не страшен, если вы не меняете номера существующих полей 03

Не следует менять типы существующих полей 04

Частые поля должны иметь номер поля до 15 включительно 05

Не рассчитывайте на стабильность бинарных и текстовых данных 06

Используйте well-known type 07

Выводы

Никогда не меняйте номера существующих полей 01

Резервируйте номера удаляемых полей 02

Protobuf не страшен, если вы не меняете номера существующих полей 03

Не следует менять типы существующих полей 04

Частые поля должны иметь номер поля до 15 включительно 05

Не рассчитывайте на стабильность бинарных и текстовых данных 06

Используйте well-known type 07

Никогда не меняйте номера существующих полей 08

OZON tech

Смотрите исходные коды по ссылке

Владислав Сидоров
vlad@ozon.ru

s.ozon.ru
/7n1Q7jM