Описание интеграционного API РНИС

> АО Группа Телематика-Один, 2020

Общее описание API

Для получения доступа к телематической информации, в том числе к живому потоку телематических данных, в РНИС используется протокол gRPC (передача Protobuf структур поверх http/2).

Технология GRPC позволяет автоматически генерировать клиентскую библиотеку для использования сервиса на основании .proto файла, описывающего структуры данных и сигнатуры методов. Генерация доступна практически под любой язык программирования.

Документация на протокол GRPC https://grpc.io/docs/guides/.

Тестовый Endpoint API доступен по адресу rnis-tm.t1-group.ru:18082.

Ниже приведено описание структур данных и методов АПИ с примерами использования на языке Golang.

Фильтрация данных

Для ограничения выборки данных во всех методах используется единая структура **DataFilter:**

```
message DataFilter {
  repeated string DeviceCode = 1;
  repeated string StateNumber = 5;
  repeated string Subsystem = 6;
  int64 DateFrom = 2;
  int64 DateTo = 3;
  int64 Limit = 4;
  repeated string ExcludeDeviceCode = 7;
}
```

Фильтрация списка необходимых ТС

Фильтрация ТС осуществляется при помощи полей:

- **DeviceCode** - массив идентификаторов БНСО

- **StateNumber** - массив госномеров ТС
- **Subsystem** - массив подсистем ТС
- **ExcludeDeviceCodes** - массив идентификаторов БНСО, исключаемых из выборки

Госномера ТС (**StateNumber**) и Подсистемы ТС (**Subsystem**) конвертируются в **DeviceCode** на стороне сервера, конвертация производится на момент осуществления запроса.

Параметры **DeviceCode**, **StateNumber** и **Subsystem** работают по принципу логического "И" - в результаты выборки попадут все устройства, перечисленные в указанных параметрах. Иными словами, допустимо использовать все три поля единовременно, в итоговый перечень **DeviceCode** для выборки попадут уникальные идентификаторы, полученные из всех трёх параметров.

Существует возможность исключить ненужные идентификаторы из итоговой выборки, для этого необходимо задать значение поля **ExcludeDeviceCode**. Данная функциональность полезна при обрывах связи, для исключения ранее загруженных объектов из выборки.

Выбор периода выгружаемых данных

Выбор периода производится параметрами **DateFrom** и **DateTo**. Формат значений - Unix Time Stamp (секунды) в часовом поясе UTC. Для запроса данных по Москве граница суток должна быть указана с 21:00 дня, предшествующего запрашиваемому периоду по 21:00 следующего дня.

Выбор полей для экспорта

Выбор полей осуществляется при помощи структуры **FieldsToggle**

```
message FieldsToggle {
  bool Position = 1;
  bool Ignition = 2;
  bool Motohours = 3;
  bool Mileage = 4;
  bool Fuel = 5;
  bool SpeedLimit = 6;
  bool Moving = 7;
  bool Ports = 8;
  bool Address = 9;
}
```

Чем меньше выбрано полей, тем меньше данных необходимо серверу выбрать из БД и в передать по сети, тем быстрее отрабатывает запрос.

Настоятельно рекомендуется задавать значение поля **Position = true ,** остальные поля оставлять со значениями по умолчанию (false), таким образом достигается максимальная производительность сервиса. Если не передавать структуру Fields в запросах по умолчанию производится выборка всех доступных полей, что негативно сказывается на производительности сервиса.

Получение данных за прошедший период

Для получения данных за прошедший период используется метод **GetObjectsDataRangeAsStream.** Данный метод возвращает телематические данные по запрошенным ТС за указанный период в формате потока.

Одно сообщение потока содержит данные по 1 ТС максимум за 1 сутки.

По завершению передачи запрошенных данных поток закрывается (на клиенте это проявляется в виде получения ошибки чтения из потока EOF).

Запрос осуществляется при помощи структуры **ObjectsDataRangeRequest**

```
message ObjectsDataRangeRequest {
  DataFilter Filter = 1;
  FieldsToggle Fields = 2;
  BoundingBox BoundingBox = 3; // в данный момент не реализовано
}
```

Метод возвращает поток объектов **ObjectData** следующей структуры.

```
message ObjectData {
  string DeviceCode = 1;
  string StateNumber = 4;
  repeated DataPoint Data = 2;
  ObjectMetadata Metadata = 3;
}
```

Один объект содержит информацию по одному DeviceCode максимум за одни сутки. Если период запроса превышает одни сутки, значение поля DateTo корректируется на стороне сервера до значения DateFrom + 1 день.

Возобновление загрузки в случае обрыва связи и ошибок

Для возобновления загрузки данных применяется следующий паттерн:

1. Выполняется запрос данных за необходимый период (как правило это одни сутки) по всем необходимым ТС
2. Производится обработка входящего потока данных, при получении и обработки каждого объекта ObjectData в состоянии клиентского приложения сохраняется информация о факте успешной обработки по каждому DeviceCode.
3. В случае обрыва связи производится повторный запрос, но в поле **ExcludeDeviceCode** добавляется перечень уже обработанных объектов, таким образом в результаты выборки попадут данные только по необработанным объектам.

Пример кода клиента

Ниже приведен пример кода загрузки данных по ТС конкретной подсистемы на языке Go.

Пример не включает логику обработки обрыва соединения (т.к. эта логика зависит от способа хранения полученных данных на стороне клиента).

```
package main

import (
  "google.golang.org/grpc"
  "time"
  "io"
  "context"
  // "externalapi/api"
  "git-02.t1-group.ru/contracts/proto-go/api"
  log "github.com/sirupsen/logrus"
)

const TIME_FORMAT = "2006-01-02 15:04:05"
```

```
func main() {

  log.Infof("# Соединение с gRPC сервисом...")
  conn, err := grpc.Dial("rnis-tm.t1-group.ru:18082", grpc.WithInsecure())
  if err != nil {
    log.Errorf("# Ошибка соединения с gRPC сервисом: %v", err)
  }

  client := api.NewAPIClient(conn)

  log.Infof("# Запрос диапазона телематики в формате потока...")

  timeFrom, _ := time.Parse(TIME_FORMAT, "2020-09-13 21:00:00")
  timeTo, _   := time.Parse(TIME_FORMAT, "2020-09-14 21:00:00")

  rangeStreamRequest := &api.ObjectsDataRangeRequest{
    Filter: &api.DataFilter{
      DateFrom:       timeFrom.Unix(),
      DateTo:       timeTo.Unix(),
      Subsystem:      []string{"kiutr"}, // для мусоровозов - garbade, на тестово
м стенде данные по garbage отсутствуют
      ExcludeDeviceCode:  []string{"10033473","404957","500459"}, // пример исклю
чения уже обработанных блоков
      DeviceCode:     []string{"10033473","404957"},  // дополнительные коды БНСО
      StateNumber:    []string{"H040PA195"},  // дополнительные госномера
    },
    Fields: &api.FieldsToggle{
      Position: true, // заправшивает только навигационную информацию
    },
  }

  rangeStream, err := client.GetObjectsDataRangeAsStream(context.Background(), ra
ngeStreamRequest)
  if err != nil {
    log.Error("# Ошибка получения потока: ", err)
  }
  for {
      object, err := rangeStream.Recv()
      if err != nil {
      if err == io.EOF {
          log.Error("# Поток закрыт, все данные переданы ", err)
            break
        } else {
            log.Fatalf("# Ошибка получения данных: %v", err)
        }
      }

    log.Infof("ObjectID: %s \t StateNumber: %s", object.DeviceCode, object.StateN
umber)
    // object.Data содержит в себе все запрошенные телематические точки по устрой
ству
    for _, point := range object.Data {
      log.Infof("\t DeviceTime: %s \tLongitude: %.8f \t Latitude: %.8f", time.Uni
x(point.DeviceTime, 0), point.Position.Longitude, point.Position.Latitude)
    }
  }
}
```

Приложения

- Код примера клиента на языке go в отдельном файле
- Proto файл с описанием структур данных и сигнатуры методов для генерации клиента