Оглавление

[Описание API сервера. 2](#_Toc421795814)

[Описание построения команд 2](#_Toc421795815)

[Авторизация (префикс А) 2](#_Toc421795816)

[Запрос авторизации: 2](#_Toc421795817)

[Автомобили (префикс B) 2](#_Toc421795818)

[Субпрефикс А – запрос списка автомобилей 2](#_Toc421795819)

[Субпрефикс B – запрос текущего состояния автомобиля 3](#_Toc421795820)

[Субпрефикс C – запрос перечня работ для автомобиля 6](#_Toc421795821)

[Субпрефикс D – запрос истории работ 7](#_Toc421795822)

[Субпрефикс E – запрос параметров авто на стоянке 7](#_Toc421795823)

[Субпрефикс F – запрос помощи от сервиса 7](#_Toc421795824)

[Субпрефикс G– отправить координаты авто в сервис 8](#_Toc421795825)

[Субпрефикс H– запрос параметров вождения 8](#_Toc421795826)

[Субпрефикс N – запрос точек остановки за указанный день 8](#_Toc421795827)

[Субпрефикс M – запрос данных для маршрутного компьютера 9](#_Toc421795828)

[Расходы (префикс С) 10](#_Toc421795829)

[Субпрефикс А – запрос списка категорий расходов 10](#_Toc421795830)

[Субпрефикс B – запрос списка расходов 10](#_Toc421795831)

[Субпрефикс С – добавить расход 11](#_Toc421795832)

[Карта магазинов (префикс D) 11](#_Toc421795833)

[Субпрефикс А – запрос списка городов 11](#_Toc421795834)

[Субпрефикс B – указание города 12](#_Toc421795835)

[Субпрефикс C – запрос категорий точек 12](#_Toc421795836)

[Субпрефикс D – запрос категорий точек 12](#_Toc421795837)

[Субпрефикс E – запрос информации о точке 13](#_Toc421795838)

# Описание API сервера.

## Описание построения команд

Команды строятся по следующим правилам:

[prefix][subprefix][data]

Префикс – один символ в UTF, обозначающий основное действие программы

Субпрефикс – один символ UTF, обозначающий поддействие для основного действия

Данные – сериализованные в JSON данные для запроса, либо – данные с разделителем (регламентируется отдельно каждым типом запроса)

Для Json сериализации существуют предикаты представления, т.е. в обычном виде сериализация происходит по имени переменной:

Public int Count{get;set;} **-> Count:0**

Для сокращения нагрузки на канал передачи данных каждая физическая переменная имеет предикат представления, который определяет, каким образом будет представлен в передаваемом сообщении тот или иной параметр:

[JsonProperty(PropertyName = "0")]

public string DID { get; set; } **-> 0:”123”**

В описании приводится структура классов для моделей передачи данных, в которых будет указан перечень предикатов для переменных

## Авторизация (префикс А)

### Запрос авторизации:

AAdemo;demo123

Где А – префикс, А – субпрефикс, demo – логин, demo123 – пароль, точка с запятой – разделитель

Ответ на авторизацию:

LERR – ошибка сборки сообщения. Может возникнуть, если потерялся разделитель

ERR – ошибка авторизации пользователя – неверно указан логин или пароль.

TOUT – таймаут. Возникает после 5 ошибок авторизации. Таймаут первоначально в 30 секунд, при следующей неверной авторизации – минута, следующая – 5 минут.

## Автомобили (префикс B)

### Субпрефикс А – запрос списка автомобилей

Пример команды: BA

Ответ сервера:

BA[data], где данные представлены в виде сериализованного списка принадлежащих пользователю автомобилей. Класс-структура для хранения данных об одном автомобиле представлена ниже, в ответе передается массив:

[JsonObject]

public class CarListBaseDataModel

{

//Идентификатор устройства, его серийный номер. На основе него в дальнейшем ведется вся обработка входящих сообщений

[JsonProperty(PropertyName = "0")]

public string DID { get; set; }

//Номер или имя автомобиля

[JsonProperty(PropertyName = "A")]

public string CarNumber { get; set; }

//Марка

[JsonProperty(PropertyName = "B")]

public string Mark { get; set; }

//Модель

[JsonProperty(PropertyName = "C")]

public string Model { get; set; }

//Текущий пробег

[JsonProperty(PropertyName = "E")]

public int CurrentDistance { get; set; }

//Пробег до ближайшей работы

[JsonProperty(PropertyName = "F")]

public int IncomeDistance { get; set; }

//название ближайшей работы

[JsonProperty(PropertyName = "G")]

public string ServiceName { get; set; }

//Дней до выполнения работы

[JsonProperty(PropertyName = "H")]

public int DaysToService { get; set; }

//Количество ошибок OBD II

[JsonProperty(PropertyName = "I")]

public int DTCErrCount { get; set; }

//Идентификатор автомобиля

[JsonProperty(PropertyName = "j")]

public int ID { get; set; }

}

### Субпрефикс B – запрос текущего состояния автомобиля

Запрос – BB

Ответ – BB[data]

Массив данных с текущими параметрами автомобиля:

[JsonObject]

public class CarStateFullModel

{

//состояние автомобиля

[JsonProperty (PropertyName = "A")]

public SCarData State

{

get { return \_state; }

set { \_state = value; }

}

//Вин-код автомобиля

[JsonProperty(PropertyName = "B")]

public string VIN { get; set; }

//Параметры OBD II

[JsonProperty(PropertyName = "C")]

public List<KVP> OBD

{

get { return \_obd; }

set { \_obd = value; }

}

//Ошибки автомобиля

[JsonProperty(PropertyName = "D")]

public List<KVP> Errors

{

get { return \_errors; }

set { \_errors = value; }

}

//Класс-модель для обозначения структуры ключ-значение

[JsonObject]

public class KVP

{

[JsonProperty(PropertyName = "E")]

public string Key { get; set; }

[JsonProperty(PropertyName = "F")]

public string Value { get; set; }

}

[JsonObject]

public class SCarData

{

//Идентификатор контроллера

[JsonProperty(PropertyName = "E")]

public string ID { get; set; }

//Данные о навигации

[JsonProperty(PropertyName = "A")]

public SNaviData Navigation

{

get { return \_navigation; }

set

{

\_navigation = value;

}

}

//Уровень сигнала сети

[JsonProperty(PropertyName = "B")]

public int GSM\_Level { get; set; }

//Напряжение питания

[JsonProperty(PropertyName = "C")]

public int PowerLevel { get; set; }

//Уровень топлива

[JsonProperty(PropertyName = "D")]

public int FuelLevel { get; set; }

//Температура контроллера

[JsonProperty(PropertyName = "F")]

public int Temperature { get; set; }

//Дата получения последнего пакета от автомобиля

[JsonProperty(PropertyName = "G")]

public DateTimeDataModel DateUpdate

{

get { return \_dateUpdate; }

set { \_dateUpdate = value; }

}

//Состояние аналоговых датчиков

[JsonProperty(PropertyName="I")]

public List<int> Sensors { get; set; }

//состояние дискретных датчиков

[JsonProperty(PropertyName="J")]

public int DisByte { get; set; }

//Ускорение

[JsonProperty(PropertyName = "K")]

public int AcsX { get; set; }

//Ускорение

[JsonProperty(PropertyName = "L")]

public int AcsY { get; set; }

//Ускорение

[JsonProperty(PropertyName = "M")]

public int AcsZ { get; set; }

//Максимальное ускорение

[JsonProperty(PropertyName = "N")]

public int AcsXMax { get; set; }

//Максимальное ускорение

[JsonProperty(PropertyName = "O")]

public int AcsYMax { get; set; }

//Максимальное ускорение

[JsonProperty(PropertyName = "P")]

public int AcsZMax { get; set; }

}

public class SNaviData

{

[JsonProperty(PropertyName = "A")]

public int Latitude { get; set; }

[JsonProperty(PropertyName = "B")]

public int Longitude { get; set; }

//кол-во спутников

[JsonProperty(PropertyName = "C")]

public int Sattelites { get; set; }

//Скорость

[JsonProperty(PropertyName = "D")]

public int Speed { get; set; }

//вектор направления

[JsonProperty(PropertyName = "E")]

public int Direction { get; set; }

}

public class DateTimeDataModel

{

//день

public int D { get; set; }

//месяц

public int M { get; set; }

//год

public int Y { get; set; }

//часы

public int hh { get; set; }

//минуты

public int mm { get; set; }

//секунды

public int ss { get; set; }

### Субпрефикс C – запрос перечня работ для автомобиля

Пример запроса:

BC[deviceID]

Где DeviceID – идентификатор устройства

Ответ передается в виде массива с списком работ:

[JsonObject]

public class WorkType

{

//идентификатор устройства

[JsonProperty(PropertyName="0")]

public string ID { get; set; }

//название работы

[JsonProperty(PropertyName = "A")]

public string Name { get; set; }

//Сколько километров до выполнения работы

[JsonProperty(PropertyName = "B")]

public int DistanceToMake { get; set; }

//Сколько дней до выполнения работы

[JsonProperty(PropertyName = "C")]

public int DaysToMake { get; set; }

//Тип работы

[JsonProperty(PropertyName="D")]

public int PresentModel { get; set; }

//периодичность выполнения в км

[JsonProperty(PropertyName = "E")]

public int PeriodicDist { get; set; }

//периодичность выполнения в мес

[JsonProperty(PropertyName = "F")]

public int PeriodicTime { get; set; }

//когда выполнялась в последний раз - дата

[JsonProperty(PropertyName = "G")]

public DateDataModel LastMakeTime { get; set; }

//когда выполнялась в последний раз - пробег

[JsonProperty(PropertyName = "H")]

public int LastMakeDist { get; set; }

}

public class DateDataModel

{

public DateDataModel(){}

public DateDataModel(DateTime dt)

{

Y = dt.Year;

M = dt.Month;

D = dt.Day;

}

public int D { get; set; }

public int M { get; set; }

public int Y { get; set; }

}

### Субпрефикс D – запрос истории работ

Построение запроса аналогично предыдущему. Ответ – массив классов:

[JsonObject]

public class CarHistoryWorkReport

{

//пробег, при котором выполнилась работа

[JsonProperty(PropertyName="B")]

public int Distance { get; set; }

//комментарий

[JsonProperty(PropertyName = "C")]

public string Comment { get; set; }

//Дата выполнения

[JsonProperty(PropertyName = "D")]

public DateDataModel Date { get; set; }

//Стоимость

[JsonProperty(PropertyName = "F")]

public int Cost { get; set; }

//Название работы

[JsonProperty(PropertyName = "G")]

public string WorkName { get; set; }

//Номер заказ-наряда

[JsonProperty(PropertyName = "H")]

public int OrderN { get; set; }

}

### Субпрефикс E – запрос параметров авто на стоянке

Запрос без параметров. Ответ:

[JsonObject]

public class CarAlarmsState

{

//Идентификатор устройства

[JsonProperty(PropertyName="A")]

public string DID { get; set; }

//Уровень удара – 0 – нет, 1- слабое, 2 – среднее, 3 - сильное

[JsonProperty(PropertyName = "B")]

public int AlarmLevel { get; set; }

//Включен ли свет 0 – нет, 1 - да

[JsonProperty(PropertyName = "C")]

public int IsLightsOn { get; set; }

//Заперта ли дверь 0 – нет, 1 - да

[JsonProperty(PropertyName = "D")]

public int IsDoorClosed { get; set; }

}

### Субпрефикс F – запрос помощи от сервиса

Пример запроса:

BF[deviceID]

Ответ:

BF[deviceID];[Status]

Статус ответа:

OK – запрос ушел в сервис

ERR – запрос по какой то причине не ушел в сервис, необходима повторная отправка

### Субпрефикс G– отправить координаты авто в сервис

Пример запроса:

BG[deviceID]

Ответ:

BG[deviceID];[Status]

Статус ответа:

OK – координаты отправлены

ERR – отправка не произошла, необходима повторная отправка

### Субпрефикс H– запрос параметров вождения

Запрос без параметров. Ответ – массив классов:

[JsonObject]

public class DrivingStyle

{

//Идентификатор устройства

[JsonProperty(PropertyName = "X")]

public string DID { get; set; }

//Текущий стиль вождения, от 0 до 100% процентов

[JsonProperty(PropertyName="A")]

public int CurrentDrivingStyle { get; set; }

//Общее значение очков вождения

[JsonProperty(PropertyName = "B")]

public int TotalDrivingScore { get; set; }

//Сегодняшнее значение очков вождения

[JsonProperty(PropertyName = "C")]

public int TodayDrivingScore { get; set; }

//Текущая экологичность вождения, от 0 до 100%

[JsonProperty(PropertyName = "D")]

public int CurrentEcoStyle { get; set; }

//общий показатель экологичности

[JsonProperty(PropertyName = "E")]

public int TotalEcoScore { get; set; }

//Дневной показатель экологичности

[JsonProperty(PropertyName = "F")]

public int TodayEcoScore { get; set; }

}

### Субпрефикс N – запрос точек остановки за указанный день

Запрос выполняется для выбора точки, от которой производить расчет. Класс запроса:

public class PointsRequestModel

{

//Дата, для которой нужно получить список

public DateDataModel Date { get; set; }

//Идентификатор устройства

public string DevID { get; set; }

}

Ответ на запрос – сериализованный массив TimeModel c точками начала движения. Даже если за день не было точек начала движения, то класс все равно придет не пустой

public class TimeModel

{

public int H { get; set; }

public int M { get; set; }

public int S { get; set; }

### Субпрефикс M – запрос данных для маршрутного компьютера

Формат запроса:

[JsonObject]

public class TripRequestModel

{

[JsonProperty(PropertyName="A")]

public DateTimeDataModel Start { get; set; }

[JsonProperty(PropertyName="B")]

public DateTimeDataModel Stop { get; set; }

[JsonProperty(PropertyName="C")]

public string devID { get; set; }

}

Формат даты и времени:

public class DateTimeDataModel

{

public int D { get; set; }

public int M { get; set; }

public int Y { get; set; }

public int hh { get; set; }

public int mm { get; set; }

public int ss { get; set; }

Указывается дата и время начала, дата и время окончания, идентификатор устройства

Ответ сервера:

[JsonObject]

public class TripAdvisorModel

{

[JsonProperty(PropertyName="X")]

public string DID { get; set; }

[JsonProperty(PropertyName = "A")]

public int CurrentDistance { get; set; }

[JsonProperty(PropertyName = "B")]

public int TripTime { get; set; }

[JsonProperty(PropertyName = "C")]

public int MedianSpeed { get; set; }

}

Сверху вниз:

* Идентификатор устройства
* Пробег в метрах
* Время в минутах
* Средняя скорость в км/ч

## Расходы (префикс С)

### Субпрефикс А – запрос списка категорий расходов

Запрос без параметров. Ответ – массив классов:

[JsonObject]

public class SpendingClassesModel

{

[JsonProperty(PropertyName="A")]

public int ID { get; set; }

[JsonProperty(PropertyName = "B")]

public string ClassName { get; set; }

}

### Субпрефикс B – запрос списка расходов

Пример запроса:

CB[deviceID]

Ответ:

[JsonObject]

public class SpendingModel

{

public SpendingModel()

{

Spends = new List<OneSpendItem>();

}

//общая сумма расходов

[JsonProperty(PropertyName="A")]

public int TotalCost { get; set; }

//Идентификатор устройства

[JsonProperty(PropertyName="B")]

public string DID { get; set; }

//Список трат

[JsonProperty(PropertyName = "C")]

public List<OneSpendItem> Spends { get; set; }

[JsonObject]

public class OneSpendItem

{

//Дата

[JsonProperty(PropertyName="D")]

public DateDataModel Date { get; set; }

//Сумма

[JsonProperty(PropertyName = "E")]

public int Sum { get; set; }

//Название

[JsonProperty(PropertyName = "F")]

public string Name { get; set; }

//Класс

[JsonProperty(PropertyName = "G")]

public int idClass { get; set; }

}

}

### Субпрефикс С – добавить расход

Пример запроса:

CB[deviceID];[data]

Где data – сериализованный класс:

[JsonObject]

public class OneSpendItem

{

//Дата

[JsonProperty(PropertyName="D")]

public DateDataModel Date { get; set; }

//Сумма

[JsonProperty(PropertyName = "E")]

public int Sum { get; set; }

//Название

[JsonProperty(PropertyName = "F")]

public string Name { get; set; }

//Класс

[JsonProperty(PropertyName = "G")]

public int idClass { get; set; }

}

## Карта магазинов (префикс D)

### Субпрефикс А – запрос списка городов

Без параметров. Ответ:

[JsonObject]

public class CityModel

{

[JsonProperty(PropertyName="A")]

public int ID { get; set; }

[JsonProperty(PropertyName = "B")]

public string CityName { get; set; }

}

### Субпрефикс B – указание города

Пример запроса:

DB[CityID]

Без ответа со стороны сервера

### Субпрефикс C – запрос категорий точек

Без параметров. Ответ:

[JsonObject]

public class PlacesClassesModel

{

[JsonProperty(PropertyName = "A")]

public int ID { get; set; }

[JsonProperty(PropertyName = "B")]

public string ClassName { get; set; }

}

### Субпрефикс D – запрос категорий точек

Запрос:

DC[idCategory];[latitude];[longitude]

idCategory – категория, если 0, то поиск по всем

Latitude, longitude – широта и долгота. В радиусе 5 км от точки будет произведен поиск

Ответ (массив данных):

[JsonObject]

class PlacesModel

{

//Идентификатор

[JsonProperty(PropertyName = "A")]

public int ID { get; set; }

//Название

[JsonProperty(PropertyName = "B")]

public string Name { get; set; }

//Адрес

[JsonProperty(PropertyName = "C")]

public string Adress { get; set; }

//Широта (поделить на 10000)

[JsonProperty(PropertyName = "D")]

public int Latitude { get; set; }

//Долгота

[JsonProperty(PropertyName = "E")]

public int Longitude { get; set; }

//Минимальный ценник

[JsonProperty(PropertyName = "F")]

public int MinPrice { get; set; }

//Рейтинг (от 0 до 100)

[JsonProperty(PropertyName = "G")]

public int Score { get; set; }

//Количество комментариев

[JsonProperty(PropertyName = "H")]

public int Comments { get; set; }

//Категория

[JsonProperty(PropertyName = "I")]

public int idCategory { get; set; }

}

### Субпрефикс E – запрос информации о точке

Запрос:

DC[idPoint]

Ответ:

[JsonObject]

public class PointDetailsModel

{

public PointDetailsModel()

{

Comments = new List<CommentModel>();

Price = new List<PriceModel>();

}

//Комментарии

[JsonProperty(PropertyName = "A")]

public List<CommentModel> Comments { get; set; }

//прайс

[JsonProperty(PropertyName = "B")]

public List<PriceModel> Price { get; set; }

[JsonObject]

public class CommentModel

{

//дата комментария

[JsonProperty(PropertyName="C")]

public DateDataModel Date { get; set; }

//Балл, от 0 до 100

[JsonProperty(PropertyName = "D")]

public int Score { get; set; }

//Текст

[JsonProperty(PropertyName = "E")]

public string Text { get; set; }

}

[JsonObject]

public class PriceModel

{

//Название

[JsonProperty(PropertyName = "F")]

public string Name { get; set; }

//Стоимость, в копейках

[JsonProperty(PropertyName = "G")]

public int Cost { get; set; }

}

}