Интеграция API
Cargo.run Логистика 2.0

Содержание

Термины и определения **5**
Назначение документа **6**
1 Общая информация **7**
1.1 API CARGO.RUN 7
1.2 Ответы сервера 7
2 Сценарии интеграции **8**
2.1 Синхронизация справочников 8
2.2 Вариант “Создание заявки в Системе” 9
2.3 Вариант “Создание заявки в учетной системе клиента” 9
3 Методы API **12**
3.1 Авторизация 12
3.1.1 Получение токена 12
3.1.2 Обновление токена 12
3.2 Общие справочники 12
3.2.1 Получение справочников 12
3.3 Справочник Водители 14
3.3.1 Добавление/редактирование водителя 14
3.3.2 Удаление водителя 14
3.3.3 Получение списка водителей 14
3.3.4 Восстановление водителя 15
3.3.5 Получение данных водителя по идентификатору 15
3.4 Справочник Машины 15
3.4.1 Добавление/редактирование машины 15
3.4.2 Удаление машины 16
3.4.3 Получение списка машин 16
3.4.4 Получение данных машины по идентификатору 16
3.4.5 Получение остановок машины 16
3.4.6 Получение времени освобождения машины 17
3.4.7 Получение текущих водителя и прицепа, последней актуальной заявки 17
3.4.8 Получение пробега за период по машине 17
3.5 Справочник Прицепы 17
3.5.1 Добавление/редактирование прицепа 17
3.5.2 Удаление прицепа 18

3.5.3 Получение списка прицепов 18
3.6 Справочник Контрагенты 18
3.6.1 Добавление/редактирование контрагента 18
3.6.2 Удаление контрагента 19
3.6.3 Получение списка контрагентов 19
3.7 Заявки 19
3.7.1 Получение адреса 19
3.7.2 Создание/редактирование заявки 20
3.7.3 Обновление заявки (только определенные поля) 21
3.7.4 Запуск заявки в работу 21
3.7.5 Получение списка заявок 21
3.7.6 Удаление заявки 22
3.7.7 Отмена заявки 23
3.7.9 Закрытие заявки вручную 23
3.7.10 Возврат активной/выполненной/отмененной заявки в черновик 24
3.7.11 Получение данных по перецепкам в заявке 24
3.7.12 Получение данных по заявке 24
3.7.13 Обновление статуса оплаты заявки 24
3.8 Трекеры 25
3.8.1 Привязка трекера к машине/прицепу. 25
3.8.2 Получение списка трекеров 26
3.8.3 Отвязка трекера к машине/прицепу. 26
3.9 Сотрудники 26
3.9.1 Получение списка сотрудников организации 26
3.10 Организация 26
3.10.1 Добавление/редактирование организации 26
3.10.2 Удаление организации 27
3.10.3 Получение списка организаций 27
3.11 Пользовательские справочники 27
3.11.1 Добавление/редактирование элемента пользовательского справочника 27
3.11.2 Удаление элемента пользовательского справочника 28
3.11.3 Получение списков пользовательских справочников 28
3.12 Заказы 28
3.12.1 Создание/редактирование заказа 28
3.12.2 Удаление заказа 31
3.12.3 Получение списка заказов 31
3.12.4 Получение заказа по идентификатору 31

3.13 ЭТРН 31
3.13.1 Получение списка титулов со статусами 31
3.13.2 Получение титула по id заявки 32
3.13.3 Смена статуса титула 33
4 Пример интеграции CARGO.RUN и 1С 34
4.1 Обновление данных заявок из CARGO.RUN в 1С 34
4.2 Варианты обновления данных 35
4.2.1 Заявка не изменилась в CARGO.RUN, не изменилась в 1С 36
4.2.2 Заявка не изменилась в CARGO.RUN, изменилась в 1С 36
4.2.3 Заявка изменилась в CARGO.RUN, не изменилась в 1С 36
4.2.4 Заявка изменилась в CARGO.RUN, изменилась в 1С 37

Термины и определения

Транспортное средство (ТС) - грузовик, машина.
Заявка (заявка на перевозку, рейс) – основная сущность в CARGO.RUN, единичная транспортировка груза от одного контрагента из точки загрузки в точку в выгрузки с указанием ТС, полуприцепа (далее просто прицепа) и водителя.
Заказ - заказ клиента без назначенных ТС, прицепа и водителя
Активный пробег - пробег от первой до последней точки заявки.
Порожний пробег - пробег от точки выгрузки предыдущей заявки до первой точки текущей заявки
Точки заявки - точки планового маршрута, могут быть точками загрузки и выгрузки.
Даты точки заявки - дата и время въезда и выезда из точки заявки. Могут быть плановыми (указывается при создании), фактическими (определение по GPS), расчетными (прогнозируется CARGO.RUN), ручными (указано пользователем при закрытии заявки в ручном режиме).
Перецепка (пересменка) - смена составляющей заявки (водитель, ТС или прицеп) в ходе выполнения заявки.
Организация - юридическое лицо транспортной компании. Если компания осуществляет деятельность с помощью нескольких юридических лиц.
Статус заявки - может принимать значение: черновик (заявка создана, но не запущена в работу), новая (запущена в работу), запланирована (запущена в работу, но ожидает выполнения текущей актуальной заявки), отменена, удалена.
Справочники - системные справочники (перечислены в 2.1 Синхронизация справочников)
Пользовательские справочники - индивидуальные справочники транспортной компании, значения справочников могут использоваться в заявках.

Назначение документа

Целью данного документа является описание способа интеграции учетной системы Клиента с CARGO.RUN и предназначена для разработчиков учетных систем.

1 Общая информация

1.1 API CARGO.RUN

REST API сервиса CARGO.RUN работает по протоколу HTTP и представляет собой набор методов, с помощью которых совершаются запросы и возвращаются ответы для каждой операции. Все ответы приходят в виде JSON структур.

Все запросы, кроме авторизации, должны быть подписаны токеном. Получение и обновление токена описано в разделе 3.1 Авторизация.

1.2 Ответы сервера

Система на вызов может дать в ответ два варианта:

1. Успешный: код “200 OK” и, в некоторых вызовах, данные (например, идентификатор объекта).
2. Ошибки:
   - 2.1. код ошибки “400” - ошибка валидации и причина;
   - 2.2. код ошибки “401” - ошибка авторизации;
   - 2.3. код ошибки “500” - ошибка на сервере.

Рекомендация: в целях оперативного выявления ошибок интеграции следует реализовать в учетной системе протоколирование ответов Системы и настройка уведомлений администратора в случае появления ошибок.

2 Сценарии интеграции

CARGO.RUN предусматривает два сценария интеграции систем:

1. Заявка создается в CARGO.RUN и передается в учетную систему Клиента.
2. Заявка создается в Учетной системе Клиента и передается в CARGO.RUN.

2.1 Синхронизация справочников

Синхронизация справочников является одинаковой для обоих сценариев интеграции.

В Системе используются следующие справочники, в которых значения поступают из учетной системы Клиента:

- “Водители”;
- “Машины”;
- “Прицепы”;
- “Контрагенты”;
- “Организации”.

Рекомендация: синхронизировать данные по этим справочникам каждую минуту.

Список справочников, значения которых поступают из CARGO.RUN в учетную систему Клиента:

- “Тип оплаты”;
- “Тип НДС”;
- “Тип груза (товара)”;
- “Бренд (марка) машины”;
- “Тип машины”;
- “Бренд (марка) прицепа”;
- “Тип прицепа”.

Этот список описан в разделе Общие справочники.

2.2 Вариант “Создание заявки в Системе”

При таком варианте сотрудники компании Клиента создают заявки в веб-интерфейсе CARGO.RUN, а Учетная система клиента соответствующими вызовами забирает информацию.

Описание вызова по получению списка заявок приведены в разделе Получение списка заявок.

Общий принцип логики вызова:

1. В учетной системе клиента создается таблица сопоставления идентификаторов CARGO.RUN и учетной системы.
2. Из CARGO.RUN запрашивается список заявок. Список может быть ограничен заявками, у которых дата изменения не старше 2-х недель.
3. Для каждой записи списка, у которой нет идентификатора в учетной системе клиента, создается новый идентификатор и записывается дата синхронизации.
4. Если у записи есть идентификатор, то анализируется дата изменения. Если она позже даты синхронизации, то данные по заявке обновляются и изменяется дата синхронизации.

Рекомендация: синхронизировать данные по заявкам раз в 1 минуту.

Внимание: во избежании конфликта при одновременной синхронизации в обе стороны рекомендуется изменять поля заявки, которые отправляются из CARGO.RUN, только в CARGO.RUN.

2.3 Вариант “Создание заявки в учетной системе клиента”

При таком варианте сотрудники компании Клиента создают заявки в учетной системе клиента, которая отправляет данные по заявкам в CARGO.RUN.

Для корректной работы интеграции требуется, чтобы в учетной системе были обязательно реализованы:

- использование геокодера для ввода адреса по всем контрольным точкам заявки (использование метода получения адреса) ;
- функция проверки на пересечение дат в заявках по одной машине.

Внимание: в случае отсутствия указанных функций невозможно гарантировать корректную работу Системы.

Описание вызова по созданию заявок приведены в разделе Создание/редактирование заявки.

Общий принцип логики вызова:

1. Учетная система клиента отправляет в CARGO.RUN запрос на создание или редактирование заявки.
2. Если запрос на создание, то в поле идентификатора заявки указывается 0. При корректном запросе CARGO.RUN возвращает ответ “200 Ок” и идентификатор заявки в CARGO.RUN, который нужно сохранить в учетной системе клиента.
3. Если запрос на редактирование, то в поле идентификатора заявки указывается соответствующий идентификатор.

Внимание: Требования к запросу на создание заявки:

1. В точке загрузки и выгрузки обязательно должен быть указан корректный адрес (поля address, city), координаты.
2. При отправке запроса должен быть заполнен минимальный набор полей.
3. Заявки должны быть не позже даты запуска пилотного проекта (или даты начала ретрансляции данных с трекеров).

Минимальный набор полей:

- тип оплаты;
- НДС;
- фиксированная цена (больше нуля);

- водитель;
- автомобиль;
- наименование груза;
- тип товара;
- адрес загрузки;
- плановые дата и время загрузки;
- адрес выгрузки;
- плановые дата и время выгрузки.

Пример реализации интеграции по данной схеме описан в разделе 4 Пример интеграции CARGO.RUN и 1С.

3 Методы API

3.1 Авторизация

3.1.1 Получение токена

```
POST /Account/GenerateToken
```

Тело запроса:

```
{
  "username": "string",
  "password": "string"
}
```

Полученный токен действителен в течение 30 минут и не требует повторного запроса при каждой операции. По истечении срока действия ключа (30 мин) необходимо отправить запрос на обновление токена

3.1.2 Обновление токена

Для обновления токена необходимо использовать метод:

```
POST /Account/RefreshToken
```

Во всех методах необходимо передавать токен. В заголовке всех запросов необходимо передавать параметры:

```
authorization: Bearer токен_полученный_при_авторизации
```

```
content-type: application/json
```

3.2 Общие справочники

3.2.1 Получение справочников

```
GET /api/catalogs/getSimple
```

Запрос на получение всех справочников

Список справочников:

`CarType`	Типы машины
`TrailerType`	Типы прицепов/полуприцепов
`CargoType`	Типы груза (товара)
`NDSType`	Типы НДС
`PaymentType`	Тип оплаты
`CarBrandType`	Бренды машин
`TrailerBrandType`	Бренды прицепов

Синхронизация справочников требуется для получения идентификаторов, которые нужно использовать при создании заявки через методы API. Все поля, которые завершаются на `*TypeId` (`paymentTypeId`, `ndsTypeId` и тд), необходимо брать из данных справочников.

При создании машины, прицепа необходимо бренд машины, прицепа также брать из общих справочников.

Пример запроса для получения всех справочников с их значениями:
`https://test.cargorun.ru/api/catalogs/getSimple`

Метод поддерживает фильтрацию в OData формате:
`?$filter=<string>&$orderBy=<string>&$top=<string>&$skip=<string>&$count=<string>&$select=<string>&$expand=<string>`

Для получения списка всех элементов справочника “Тип машины” необходимо выполнить запрос:
`/api/catalogs/getSimple?$filter=propertyName eq 'CarType'`

Для получения элементов других типов справочников необходимо подставлять в запрос их название (`PaymentType`, `CargoType` и т.д.).

3.3 Справочник Водители

3.3.1 Добавление/редактирование водителя

```
POST /api/Driver/Apply
```

Тело запроса:

```
{
  "id": 0,                    // если указывается 0, то создается новый водитель
  "firstName": "string",      // Имя
  "lastName": "string",       // Фамилия
  "patronymic": "string",     // Отчество
  "phoneNumber": "string",    // номер телефона в формате +79001234567
  "isDeleted": true,          // признак уволенного водителя
  "comment": "string",        // текстовый комментарий
  "needsShiftChangeComment": "string", // комментарий о потребности в пересменке
  "needsUrgentShiftChange": true,   // комментарий о срочной потребности в
пересменке
  "transportColumnId": 0      // идентификатор транспортной колонны

}
```

При добавлении нового водителя необходимо в поле id указывать 0. При обновлении данных по водителю указывать идентификатор существующей записи. Номер телефона водителя необходимо вводить в формате “+79001234567”.

3.3.2 Удаление водителя

```
POST /api/Driver/Delete
```

При вызове метода водитель помечается как удаленный.

3.3.3 Получение списка водителей

```
GET /api/Driver/GetList
```

Метод поддерживает фильтрацию в OData формате.

```
?checkOnline=false&$filter=<string>&$orderBy=<string>&$top=<string>&$skip=<string>&
$count=<string>&$select=<string>&$expand=<string>
```

3.3.4 Восстановление водителя

```
POST /api/Driver/Restore
```

Тело запроса:

```
{
  "id": 0                       // идентификатор водителя для удаления
}
```

3.3.5 Получение данных водителя по идентификатору

```
GET /api/Driver/GetForEdit
```

Идентификатор передается параметром, пример:

```
GET https://test.cargorun.ru/api/Driver/GetForEdit?Id=5
```

3.4 Справочник Машины

3.4.1 Добавление/редактирование машины

```
POST /api/Car/Apply
```

Тело запроса:

```
{
  "id": 0,                      // если указывается 0, то создается новая машина
  "logistId": 0,                // Идентификатор ответственного логиста
  "typeId": 0,                  // Идентификатор типа машины (тягач, самосвал)
  "number": "string",           // Госномер машины
  "brandTypeId": 0,             // Идентификатор типа бренда машины (КАМАЗ, MAN)
  "isDeleted": true,            // признак удаленной машины
  "comment": "string"           // текстовый комментарий
  "mechanic": {                 // указание механика в машине
    "id": 0,                    // идентификатор механика, если новый, то 0
    "phoneNumber": "string",    // Номер телефона ответственного за машину механика
    "name": "string"            // ФИО ответственного за машину механика
  },
  "needsMaintenanceComment": "string", // комментарий о потребностях в ТО/ремонте
  "needsUrgentMaintenance": true    // комментарий о срочных потребностях в ТО/ремонте
  "transportColumnId": 0        // идентификатор колонны
```

```
}
```

При добавлении необходимо в поле id указывать 0, при обновлении номер существующей записи.
Получение списка всех элементов справочников “Тип машины”, “Бренд машины” выполняется через запрос GetSimple (3.2.1 Получение справочников).

3.4.2 Удаление машины

```
POST /api/Car/Delete
```

Тело запроса:

```
{
  "id": 0                          // идентификатор машины для удаления
}
```

3.4.3 Получение списка машин

```
GET /api/Car/GetList
```

Метод поддерживает фильтрацию в OData формате. Параметры:

```
?$filter=<string>&$orderBy=<string>&$top=<string>&$skip=<string>&$count=<string>&$select=<string>&$expand=<string>
```

3.4.4 Получение данных машины по идентификатору

```
GET /api/Car/GetForEdit
```

Идентификатор передается параметром, пример:

```
GET https://test.cargorun.ru/api/Car/GetForEdit?Id=5
```

3.4.5 Получение остановок машины

```
GET /api/Car/GetStops
```

Метод поддерживает фильтрацию. Для оптимального запроса необходимо фильтровать по CarID и Date.

```
?$filter=<string>&$orderBy=<string>&$top=<string>&$skip=<string>&$count=<string>&$select=<string>&$expand=<string>
```

3.4.6 Получение времени освобождения машины

```
GET /api/Car/GetAvailabilityStatus/{carId}
```

3.4.7 Получение текущих водителя и прицепа, последней актуальной заявки

```
GET /api/Car/GetCarLastBidInfo/{carId}
```

3.4.8 Получение пробега за период по машине

```
GET /api/Car/GetCarMileageForPeriod
```

Параметры запроса:

```
?CarId=<long>&Start=<dateTime>&End=<dateTime>
```

Формат даты: `YYYY-MM-DDTHH:mm:ss`

Пример запроса:

```
/api/car/getcarmileageforperiod?carid=694385&start=2025-10-31T21:00:00&end=2025-11-17T20:59:59
```

3.5 Справочник Прицепы

3.5.1 Добавление/редактирование прицепа

```
POST /api/Trailer/Apply
```

Тело запроса:

```
{
      "id": 0,                    // если указывается 0, то создается новый прицеп
      "typeId": 0,                // идентификатор типа прицепа
      "number": "string",         // госномер прицепа
      "brandTypeId": 0            // идентификатор бренда
}
```

При добавлении нового прицепа необходимо в поле id указывать 0, при обновлении - идентификатор существующего.

3.5.2 Удаление прицепа

```
POST /api/Trailer/Delete
```

Тело запроса:

```
{
  "id": 0                        // идентификатор прицепа для удаления
}
```

3.5.3 Получение списка прицепов

```
GET /api/Trailer/GetList
```

Метод поддерживает фильтрацию в OData формате.

```
?$filter=<string>&$orderBy=<string>&$top=<string>&$skip=<string>&$count=<string>&$select=<string>&$expand=<string>
```

3.6 Справочник Контрагенты

3.6.1 Добавление/редактирование контрагента

```
POST /api/CargoOwnerDictionary/Apply
```

Тело запроса:

```
{
      "id": 0,                        // если указывается 0, то создается новый контрагент
      "name": "string",               // наименование контрагента
      "inn": "string",                // ИНН контрагента
      "kpp": "string",                // КПП контрагента
      "comment": "string",            // Комментарий по контрагенту
      "noteForDrivers": "string",     // Заметки для водителя
      "analyzeStrictOffsetHours": 0,// Период анализа посещения точек в часах (от 12 до 96)
      "status": "None",               // Статус контрагента
      "contactPerson": {              // Информация по контактному лицу
        "firstName": "string",
        "lastName": "string",
        "patronymic": "string",
        "phoneNumber": "string",
        "email": "string"
       }
```

```
}
```

При добавлении необходимо в поле id указывать 0, при обновлении номер существующей записи.
Возможные статусы контрагента:

`"None"`	Без свойств
`"Insolvent"`	Неплатежеспособен
`"Priority"`	Приоритетный
`"Suspended"`	Приостановлен

3.6.2 Удаление контрагента

```
POST /api/CargoOwnerDictionary/Delete
```

Тело запроса:

```
{
  "id": 0                          // идентификатор контрагента для удаления
}
```

3.6.3 Получение списка контрагентов

```
GET /api/CargoOwnerDictionary/Get
```

Метод поддерживает фильтрацию в OData формате.

```
?$filter=<string>&$orderBy=<string>&$top=<string>&$skip=<string>&$count=<string>&$s
elect=<string>&$expand=<string>
```

3.7 Заявки

3.7.1 Получение адреса

```
POST /api/Map/GetAddresses
```

Тело запроса:

```
{
  "address": "<string>"                        // адрес в виде текста
```

```
}
```

Для того, чтобы получить структурированную информацию об адресе точки загрузки/выгрузки (регион, район, населенный пункт, координаты точки) в формате Яндекса необходимо запустить данный метод. На вход передается адрес точки загрузки/выгрузки в виде текстовой строки.

3.7.2 Создание/редактирование заявки

```
POST /api/TruckingBids/Apply
```

При добавлении необходимо в поле id указывать 0, при обновлении номер существующей записи.
В ответе возвращается идентификатор заявки CARGO.RUN, статус построения маршрута.

При создании заявки поля, которые завершаются на *TypeId (`paymentTypeId`, `ndsTypeId` и тд), необходимо надо брать из общих справочников

Пример тела запроса создания заявки: API-TruckingBids-Apply

Статусы заявок:

`"New"`	Черновик
`"Started"`	Начата (запущена в работу)
`"Planned"`	Запланирована (запущена в работу)
`"Done"`	Выполнена
`"Canceled"`	Отменена

Для указания идентификатора 1С используется поле:
`"externalId": "string"`

Для указания порожней заявки (заявки без груза) используются поля:

```
  "emptyMileageBid": {
    "logisticReasonTypeId": 0,
```

```
    "technicalReason": "string"
```

Информация по грузу, контрагенту не указывается, стоимость = 1.

3.7.3 Обновление заявки (только определенные поля)

`POST /api/TruckingBids/Patch`

Если необходимо изменить только определенные поля, то необходимо указывать только их.

Можно изменить любое поле в заявке кроме вложенных объектов. Если нужно изменить точку маршрута, то нужно передавать все точки маршрута помимо той, которые нужно изменить.

3.7.4 Запуск заявки в работу

`POST /api/TruckingBids/SetStatus`

Тело запроса:

```
{
      "bidId": 0,
      "status": "Started"
}
```

3.7.5 Получение списка заявок

`GET /api/bids/GetListForExternal`

Результат будет представлять ответ с HTTP-статусом 200 (Успех). Метод вернет массив объектов, пример ответа.

У каждой заявки в CARGO.RUN есть «Дата обновления» (`updatedAt`) – это дата последнего обновления по заявке. В синхронизации CARGO.RUN и 1С мы используем эту дату. Запрашиваем заявки, у которых эта дата больше начала предыдущего дня. И полученный список заявок загружаем в 1С. При сохранении заявки в 1С записываем эту дату обновления и при следующей синхронизации сравниваем дату обновления в CARGO.RUN и 1С. Если они различаются, значит заявка изменилась и перезаполняем её в 1С.

Пример:

```
http://app.cargorun.ru/api/bids/GetListForExternal?$filter=updatedAt ge
2024-01-23T21:00:00Z and updatedAt le 2024-01-25T20:59:59Z and createdAt ge
2022-09-30T21:00:00Z&$top=50&$orderby=updatedAt&$skip=0
```

Получаем список всех заявок:

- дата обновления которых больше 24.01.2024 00:00 МСК (23.01.2024 21:00:00 UTC);

- дата обновления которых меньше 25.01.2024 23:59 МСК (25.01.2024 20:59:59 UTC);

- дата создания больше 30.09.2022 (если необходимо ограничить по дате создания заявок);

- первые 50, начиная с 0;

- сортировка по дате обновления.

3.7.6 Удаление заявки

Для того, чтобы удалить отмененную заявку, ее необходимо вернуть в черновик и удалить.

Возврат активной/выполненной/отмененной заявки в статус черновика:

```
POST /api/truckingbids/revert
```

Тело запроса:

```
{
      "bidId": 0
}
```

Удаление заявки:

```
POST /api/bids/delete
```

Тело запроса:

```
{
      "bidId": 0
}
```

3.7.7 Отмена заявки

```
POST /api/bids/cancel
```

Тело запроса:

```
{
      "bidId": 0
}
```

3.7.8 Восстановление удаленной заявки

```
POST /api/bids/Restore
```

Тело запроса:

```
{
      "bidId": 0
}
```

3.7.9 Закрытие заявки вручную

```
POST /api/truckingbids/forceComplete
```

Тело запроса:

```
{
  "bidId": 0,                 // Идентификатор заявки
  "reason": "string",         // Причина ручного закрытия
  "mileage": 0,               // Фактический километраж
  "useOdometerMileage": true, // Использовать пробег по одометру
  "bidPoints": [              // Точки заявки, массив
    {
      "id": 0,                // Идентификатор точки
      "enteredAtByLogist": "2025-05-07T07:15:04.555Z",      // Фактическая дата
въезда
      "loadUnloadedAtByLogist": "2025-05-07T07:15:04.555Z",// Фактическая дата
выезда
      "loadUnloadStatus": "AtLoading"
    }
  ]
}
```

3.7.10 Возврат активной/выполненной/отмененной заявки в черновик

```
POST /api/TruckingBids/Revert
```

Тело запроса:

```
{
      "bidId": 0
}
```

3.7.11 Получение данных по перецепкам в заявке

```
GET /api/bids/GetListForExternal
```

Для определения была ли перецепка в заявке нужно в проверять параметр `hasItemsChange` в ответе метода `GetListForExternal`. Если перецепка была то значение `true`, если не было - `false`.

```
GET /api/TruckingBids/GetBidItemsChanges?bidId=<long>
```

Метод возвращает массив значений, где сначала идут первоначальные данные, потом новые данные.

3.7.12 Получение данных по заявке

```
GET /api/bids/Get/{bidId}
```

3.7.13 Обновление статуса оплаты заявки

В ответы на запросы

```
GET /api/bids/get/{bidId},
```

```
GET /api/truckingbids/get/{bidId}
```

добавлен элемент:

```
"bidPayment": {
    "planPaymentDate": "2023-12-31",  // Плановая дата платежа
    "paymentStatus": "PartiallyPaid", // Статус
    "remainingPayment": 5000.50,      // Остаток к оплате (если есть)
    "updatedAt": "2023-12-15T14:30:00Z" // Дата обновления
```

```
  }
```

В тело запроса метода редактирования заявки (3.7.2 Создание/редактирование заявки)

```
POST /api/truckingBids/apply
```

добавлен элемент

```
"payment": {
    "planPaymentDate": "2024-02-20T00:00:00Z",  // Обязательное поле
    "paymentStatus": "NotPaid",                 // Обязательное поле, статус
    "remainingPayment": null              // Опциональное, если статус
PartiallyPaid
  }
```

Статусы оплаты:

`"NotPaid"`	Не оплачено
`"Paid"`	Оплачено
`"PartiallyPaid"`	Частично оплачено
`"Expired"`	Просрочено

3.8 Трекеры

3.8.1 Привязка трекера к машине/прицепу.

```
POST /api/Trackers/Attach
```

Тело запроса:

```
{
      "id": 0,                      // Идентификатор трекера
      "entityId": 0,                // Идентификатор объекта (прицеп, машина)
      "type": "Car"                 // Тип объекта (“Trailer”, “Car”)
}
```

Для привязки необходимо указывать номер id трекера и машины/прицепа.

Чтобы привязать трекер к машине необходимо выполнить поиск по номеру `/api/Trackers/Get?$filter=deviceNumber eq '1234567890'` и его id добавить в метод `Attach`.

3.8.2 Получение списка трекеров

```
GET /api/Trackers/Get
```

Метод поддерживает фильтрацию в OData формате.

```
?$filter=<string>&$orderBy=<string>&$top=<string>&$skip=<string>&$count=<string>&$s
elect=<string>&$expand=<string>
```

3.8.3 Отвязка трекера к машине/прицепу.

```
POST /api/Trackers/Attach
```

Тело запроса:

```
{
      "id": 0,                      // Идентификатор трекера
      "entityId": null,             // Идентификатор объекта (прицеп, машина)
      "type": "Car"                 // Тип объекта (“Trailer”, “Car”)
}
```

Для отвязки трекера необходимо указывать номер id трекера, идентификатор машины/прицепа со значением `null`.

3.9 Сотрудники

3.9.1 Получение списка сотрудников организации

```
GET /api/Employees/GetNamesList
```

Метод поддерживает фильтрацию в OData формате.

```
?$filter=<string>&$orderBy=<string>&$top=<string>&$skip=<string>&$count=<string>&$s
elect=<string>&$expand=<string>
```

3.10 Организация

3.10.1 Добавление/редактирование организации

```
POST /api/LegalPersons/Apply
```

Тело запроса:

```
{
      "id": 0,                        // При создании нового указывать 0, при
      редактировании указывать id существующего
      "name": "string",               // Наименование организации
      "inn": "string"                 // ИНН организации
}
```

3.10.2 Удаление организации

```
POST /api/LegalPersons/Delete
```

Тело запроса:

```
{
      "id": 0                         // Id удаляемой организации

}
```

3.10.3 Получение списка организаций

```
GET /api/LegalPersons/GetList
```

Метод поддерживает фильтрацию в OData формате.

```
?$filter=<string>&$orderBy=<string>&$top=<string>&$skip=<string>&$count=<string>&$s
elect=<string>&$expand=<string>
```

3.11 Пользовательские справочники

3.11.1 Добавление/редактирование элемента пользовательского справочника

```
POST /api/Catalogs/ApplyOrganizationItem
```

Тело запроса:

```
{
      "id": 0,          // идентификатор добавляемого элемента
      "catalogId": 0,   // идентификатор справочника в который добавляется новый
элемент
      "displayName": "string" //название элемента справочника
}
```

При добавлении необходимо в поле id указывать 0, при обновлении номер существующей записи.

3.11.2 Удаление элемента пользовательского справочника

```
POST /api/Catalogs/DeleteItem
```

Тело запроса:

```
{
      "id": 0     //идентификатор удаляемого элемента пользовательского справочника
}
```

3.11.3 Получение списков пользовательских справочников

```
GET /api/catalogs/getUserEntityOptions
```

Метод поддерживает фильтрацию в OData формате.

```
?$filter=<string>&$orderBy=<string>&$top=<string>&$skip=<string>&$count=<string>&$s
elect=<string>&$expand=<string>
```

3.12 Заказы

3.12.1 Создание/редактирование заказа

```
POST /api/DistributionBids/Apply
```

Тело запроса аналогично методу создания заявки (3.7.2 Создание/редактирование заявки), за исключением указания водителя, машины, прицепа.

Пример запроса на создание заказа:

Тело запроса

```
{
    "items": [
        {
            "counterpartyId": 3742683, //ИД контрагента
            "price": "100000", //сумма по заявке
            "paymentTypeId": 176, //тип оплаты (наличный/безналичный)
            "ndsTypeId": 173, //ставка НДС
            "cargos": [ //тут информацию о грузе
                {
                    "name": "Тут наименование груза (любой текст)",
                    "typeId": 163, //тип груза из справочника Каргоран
(необязательно)
                    "comment": "Это комментарий к грузу",
                    "temperatureMinimum": "-4",
                    "temperatureMaximum": "-2"
                }
            ],
            "comment": "Это комментарий к заказу",
            "contractNumber": "Договор №456", //номер договора с контрагентом
            "bidPoints": [ //информация по точкам. порядок заполнения аналогичен
порядку создания заявок
                {
                    "order": 0,
                    "type": "LoadPoint",
                    "planEnterDate": "2021-04-15 09:07",
                    "geozone": {
                        "location": {
                            "type": "Point",
                            "coordinates": [
                                37.59590148925782,
                                55.72556335763931
                            ]
                        },
                        "address": "Россия, Москва, улица Крымский Вал, 9с52",
                        "city": "Москва",
                        "state": "Москва",
                        "county": null,
                        "street": "улица Крымский Вал",
                        "houseNumber": "9с52",
                        "federalDistrict": "Центральный федеральный округ",
                        "radius": 0
```

```
            },
            "cargoOwnerDictionaryItemId": null,
            "contactPerson": {}
        },
        {
            "order": 1,
            "type": "UnloadPoint",
            "planEnterDate": "2021-04-16 09:07",
            "geozone": {
                "location": {
                    "type": "Point",
                    "coordinates": [
                        39.74578857421876,
                        54.62615829892253
                    ]
                },
                "address": "Россия, Рязань, улица Радищева",
                "city": "Рязань",
                "state": "Рязанская область",
                "county": "городской округ Рязань",
                "street": "улица Радищева",
                "houseNumber": null,
                "federalDistrict": "Центральный федеральный округ",
                "radius": 0
            },
            "cargoOwnerDictionaryItemId": null,
            "contactPerson": {}
        },
        {
            "order": 2,
            "type": "UnloadPoint",
            "planEnterDate": "2021-04-17 09:07",
            "geozone": {
                "location": {
                    "type": "Point",
                    "coordinates": [
                        37.976303100585945,
                        55.60705698351368
                    ]
                },
                "address": "Россия, Московская область, городской округ
Люберцы, рабочий посёлок Октябрьский",
                "city": "рабочий посёлок Октябрьский",
```

```
                        "state": "Московская область",
                        "county": "городской округ Люберцы",
                        "street": null,
                        "houseNumber": null,
                        "federalDistrict": "Центральный федеральный округ",
                        "radius": 0
                    },
                    "cargoOwnerDictionaryItemId": null,
                    "contactPerson": {},
                    "extendedProperties": []
                }
            ],
            "contactPerson": { - тут информация о контактном лице контрагента
                "name": "ФИО контактного лица",
                "phoneNumber": "+79516234870"
            }
        }
    ]
}
```

Тело ответа

```
{
      "id": 4140742 //в ответе возвращается идентификатор созданного заказа
      }
```

3.12.2 Удаление заказа

```
POST /api/DistributionBids/Delete
```

Удаление заказа, если по нему не сделана заявка.

3.12.3 Получение списка заказов

```
GET /api/DistributionBids/GetList
```

Метод поддерживает фильтрацию в OData формате.

3.12.4 Получение заказа по идентификатору

```
GET /api/DistributionBids/Get/{id}
```

3.13 ЭТРН

3.13.1 Получение списка титулов со статусами

```
GET /api/1c/Bills/GetTitles
```

Метод поддерживает фильтрацию в OData формате.

Тело ответа:

```
[
 {
   "id": 0, - Id титула в cargo.run
   "type": "T1", - тип титула
   "status": "Received",  - статус титула
   "statusDate": "2024-05-31T14:31:49.558176+00:00", - дата проставления статуса
   "bill": {
     "id": 0, - Id ЭТрН в cargo.run
     "gisId": "b03c6b63-88e0-4e51-9650-bab89b13141e", -- GUID ЭТрН в ГИС ЭПД
     "bid": {
       "id": 0, - Id заявки в cargo.run
       "externalId": "0" - Id заявки во внешней системе
     }
   }
 }
]
```

Статусы ЭТРН:

"Received"	Титул получен Логистикой
"Accepted"	Титул принят (водителем или внешней системой)
"Declined"	Титул не принят (не используется)
"SentToCrEBill"	Титул передан на отправку

3.13.2 Получение титула по id заявки

```
GET
/api/1c/Bills/GetTitle?titleId={titleId}&bidId={bidId}&bidExternalId={bidExternalId
}
```

Нужно указать id титула (titleId) + id заявки в CARGO.RUN (bidId), либо id титула (titleId) + id заявки из 1с (externalId).

Тело ответа:

```
{
"remarks": {
   "factEnteredAt": "2023-07-24T07:55:11.968Z", - Фактическая дата и время въезда на точки загрузки/выгрузки (T2 и T4)
   "factLeftAt": "2023-07-24T07:55:11.968Z", - Фактическая дата и время выезда с точки загрузки/выгрузки (T2 и T4)
   "factLoadUnloadAt": "2023-07-24T07:55:11.968Z", - Фактическая дата и время загрузки/выгрузки (T2 и T4)
   "cargoConditionNotes": "string", - Замечания по состоянию груза (T2 и T4)
   "cargoWeightNotes": "string", - Замечания о массе груза (T2)
   "cargoPlacesNotes": "string", - Замечания по кол-ву мест (T2)
   "loadingUnloadingNotes": "string", - Замечания о погрузочных/разгрузочных работах (T2 и T4)
   "note": "string" - Другие замечания (T2 и T4)
 },
 "id": 21,
 "type": "T1",
 "status": "Received",
 "statusDate": "2024-05-31T14:31:49.558176+00:00"
}
```

3.13.3 Смена статуса титула

```
POST  /api/1c/Bills/ApplyStatus
```

Пример запроса на смену титула:

Тело запроса

```
{
 "bidExternalId": "0", -- обязателен, если не указан bidId
 "bidId": 0, -- обязателен, если не указан bidExternalId
 "id": 0,
 "status": "Accepted"
}
```

4 Пример интеграции CARGO.RUN и 1С

Система: CARGO.RUN
Учетная система Клиента: 1С

В 1С необходимо реализовать:

- создание дополнительной таблицы для интеграции;
- при создании, обновлении справочника (водитель, машина, прицеп, контрагент), заявки записывать в дополнительную таблицу ссылку на неё;
- 1 раз в 15 минут отправлять все созданные/измененные в этот период справочники, заявки из этой таблицы из 1С в CARGO.RUN;
- при создании элемента справочника, заявки необходимо записывать в 1С id из CARGO.RUN. При следующем обновлении необходимо указывать этот id.
- при создании, обновлении элемента справочника, заявки необходимо записывать в 1С дату обновления;
- в 1С у каждой записи справочника, заявки должны быть 2 даты:
    - дата обновления в CARGO.RUN.
    - дата обновления в 1С.

4.1 Обновление данных заявок из CARGO.RUN в 1С

Если необходимо изменять данные заявок в CARGO.RUN (например, изменять адреса точек загрузки/выгрузки), то из 1С необходимо запрашивать изменения. Это выполняется следующим образом:

- в 1С необходимо записывать дату обновления заявки (при создании, обновлении);
- запрашивать обновления по заявкам в CARGO.RUN (список заявок с датой обновления - `UpdatedAt`) - `/api/bids/GetListForExternal`.
  Необходимо выбирать выполненные заявки (`status = 'Done'`)
  Фактический километраж по заявке - `factMileage`

По точкам загрузки/выгрузки:
Дата въезда в геозону точки - `bidPoints -> autoEnteredAt`

Дата выезда в геозону точки - `bidPoints -> autoLeavedAt`

Пример запроса по выполненным заявкам больше определенной даты:

```
/api/bids/GetListForExternal?$filter=Status eq 'Done' and updatedAt gt
2019-11-20T06:00:00Z
```

- если в CARGO.RUN дата обновления заявки более поздняя, чем в 1С, то необходимо обновить данные заявки из CARGO.RUN.
- При получении данных по заявке пользовательские поля описываются в массиве `"extendedProperties"` который имеет вид:

```
"extendedProperties": [
      {
          "propertyName": "Due_date", //название пользовательского поля
          "value": "12" //значение указанное в пользовательском поле
      }
  ]
```

- При получении данных по заявке пользовательские справочники описываются в массиве `"typeOptions "` и имеет вид:
-

```
"typeOptions": [
      {
           "id": 3795270, // идентификатор справочника в организации
          "entityOptionId": 3794317 // идентификатор справочника в CARGO.RUN
      }
 ]
```

4.2 Варианты обновления данных

Возможны четыре варианта ситуации при обновлении данных из CARGO.RUN в 1С:

1. Заявка не изменилась в CARGO.RUN, не изменилась в 1С.
2. Заявка не изменилась в CARGO.RUN, изменилась в 1С.
3. Заявка изменилась в CARGO.RUN, не изменилась в 1С.
4. Заявка изменилась в CARGO.RUN, изменилась в 1С.

Ниже приведено описание каждого варианта.

4.2.1 Заявка не изменилась в CARGO.RUN, не изменилась в 1С

Дата последней синхронизации	12.11.2019 10:00
Дата обновления заявки в CARGO.RUN	11.11.2019 09:00
Дата обновления заявки в 1С	12.11.2019 09:10
Запуск синхронизации	12.11.2019 10:15

В результате:

1. Дата последней синхронизации > Дата обновления заявки в CARGO.RUN.
2. Дата последней синхронизации > Дата обновления заявки в 1С.

Итого: Обновление данных не требуется.

4.2.2 Заявка не изменилась в CARGO.RUN, изменилась в 1С

Дата последней синхронизации	12.11.2019 10:00
Дата обновления заявки в CARGO.RUN	11.11.2019 16:00
Дата обновления заявки в 1С	12.11.2019 10:05
Запуск синхронизации	12.11.2019 10:15

В результате:

1. Дата последней синхронизации > Дата обновления заявки в CARGO.RUN.
2. Дата обновления заявки в 1С > Дата последней синхронизации

Итого: Необходимо обновить заявку в CARGO.RUN из 1С

4.2.3 Заявка изменилась в CARGO.RUN, не изменилась в 1С

Дата последней синхронизации	12.11.2019 10:00
Дата обновления заявки в CARGO.RUN	12.11.2019 10:09

Дата обновления заявки в 1С	10.11.2019 15:00
Запуск синхронизации	12.11.2019 10:15

В результате:

1. Дата обновления заявки в CARGO.RUN > Дата последней синхронизации.
2. Дата последней синхронизации > Дата обновления заявки в 1С.

Итого: Необходимо обновить заявку в 1С из CARGO.RUN

4.2.4 Заявка изменилась в CARGO.RUN, изменилась в 1С

Дата последней синхронизации	12.11.2019 10:00
Дата обновления заявки в CARGO.RUN	12.11.2019 10:10
Дата обновления заявки в 1С	12.11.2019 10:05
Запуск синхронизации	12.11.2019 10:15

В результате:

1. Дата обновления заявки в CARGO.RUN > Дата последней синхронизации.
2. Дата обновления заявки в 1С > Дата последней синхронизации.

Итого: Заявка обновилась в CARGO.RUN, и в 1С, конфликт обновления.

Предполагается два варианта разрешения конфликта: автоматический и ручной.

При автоматическом режиме разрешения конфликтов необходимо добавить приоритетную сторону. Это может быть 1С или CARGO.RUN.
Если выбран 1С приоритетной стороной, то при конфликте данные в CARGO.RUN обновляется из 1С.
Если выбран CARGO.RUN приоритетной стороной, то при конфликте данные в 1С обновляется из CARGO.RUN.

При ручном режиме разрешения конфликтов необходимо реализовать в 1С интерфейс для просмотра конфликтных данных и выбор, с какой из систем обновлять данные.