

Интеграция API Cargo.run



Содержание

Термины и определения	4
Назначение документа	5
1 Общая информация	6
1.1 Описание API	6
1.2 Полный список вызовов API	6
1.3 Полный список вызовов API	6
2 Варианты интеграции	7
2.1 Синхронизация справочников	7
2.2 Вариант “Создание заявки в Системе”	7
2.3 Вариант “Создание заявки в учетной системе клиента”	8
3 Запросы API	11
3.1 Авторизация	11
3.1.1 Получение токена	11
3.1.2 Обновление токена	11
3.2 Общие справочники	11
3.2.1 Получение справочников	11
3.3 Справочник Водители	12
3.3.1 Добавление/редактирование водителя	12
3.3.2 Удаление водителя	12
3.3.3 Получение списка водителей	12
3.4 Справочник Машины	12
3.4.1 Добавление/редактирование машины	12
3.4.2 Удаление машины	13
3.4.3 Получение списка машин	13
3.5 Справочник Прицепы	13
3.5.1 Добавление/редактирование прицепа	13
3.5.2 Удаление прицепа	13
3.5.3 Получение списка прицепов	13

3.6	Справочник Контрагенты	13
3.6.1	Добавление/редактирование контрагента	13
3.6.2	Удаление контрагента	13
3.6.3	Получение списка контрагентов	14
3.7	Заявки	14
3.7.1	Получение адреса	14
3.7.2	Создание/редактирование заявки	14
3.7.3	Обновление заявки (только определенные поля)	14
3.7.4	Запуск заявки в работу	14
3.7.5	Получение списка заявок	15
3.7.6	Удаление заявки	15
3.7.7	Отмена заявки	15
4	Пример интеграции Cargo.Run и 1С	16
4.1	Обновление данных заявок из Cargo.Run в 1С	16
4.2	Варианты обновления данных	17
4.2.1	Заявка не изменилась в Cargo.Run, не изменилась в 1С	17
4.2.2	Заявка не изменилась в Cargo.Run, изменилась в 1С	17
4.2.3	Заявка изменилась в Cargo.Run, не изменилась в 1С	18
4.2.4	Заявка изменилась в Cargo.Run, изменилась в 1С	18

Термины и определения

Заявка

Активный пробег

Порожний пробег

Логист

Рубль/километр

Трекер

Координаты

Расчетное время прибытия

Плановое нахождение

Клиент

Токен

API

Swagger

Учетная система Клиента - 1С, SAP

Администратор

Назначение документа

Система Cargo.Run (далее Система) предназначена для автоматизации операционной логистической деятельности.

API интерфейс используется для того, чтобы интегрировать возможности цифровой логистической платформы Cargo.Run и учетной системы Клиента.

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

1 Общая информация

1.1 Описание API

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 Полный список вызовов API

Список API методов (swagger) с доступен по адресу:

<https://test.cargorun.ru/swagger/index.html>

Для доступа следует воспользоваться учетной записью:

- логин: cargorun
- пароль: cargorun_api123

1.3 Полный список вызовов API

Система на вызов может дать в ответ два варианта:

1. Успешный: код “200 ОК” 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 Варианты интеграции

Система предусматривает два сценария интеграции систем:

1. Заявка создается в Системе и передается в учетную систему Клиента.
2. Заявка создается в Учетной системе Клиента и передается в Систему.

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”](#).

Рекомендация: синхронизировать данные по этим справочникам каждые 15 минут.

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 [Общие справочники](#).

2.2 Вариант “Создание заявки в Системе”

При таком варианте сотрудники компании Клиента создают заявки в 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 2х 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 Запросы API

3.1 Авторизация

3.1.1 Получение токена

/Account/GenerateToken

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

Обратите внимание, что 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:

/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

/api/Catalogs/Get

Запрос на получение всех справочников

Список справочников:

- CarType - Типы машины

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

Пример запроса для получения всех справочников с их значениями:

```
https://test.cargorun.ru/api/catalogs/get?$select=id,type,propertyName,displayName&$expand=entityOptions($select=id,displayName),items($filter=isDeleted%20eq%20false;$expand=itemEntityOptions($expand=entityOption($select=id,propertyName,displayName,valueCatalogType,valueClrType);$select=id);$select=id,propertyName,displayName)&$orderBy=id%20desc
```

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

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

Для получения списка всех элементов справочника “Тип машины” необходимо выполнить запрос:

```
/api/catalogs/get?$filter=type eq 'CarType'&$expand=items($filter=isDeleted eq false)
```

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

Ссылка на номера id справочников -

https://docs.google.com/spreadsheets/d/1QbLSJIMafCtL8-t6NxClo_NCC4iuPbRP52B7HZ-tgIg/edit#gid=0

3.3 Справочник Водители

3.3.1 Добавление/редактирование водителя

/api/Driver/Apply

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

Номер телефона водителя необходимо вводить в формате “+79001234567”.

3.3.2 Удаление водителя

/api/Driver/Delete

3.3.3 Получение списка водителей

/api/Driver/GetList

3.4 Справочник Машины

3.4.1 Добавление/редактирование машины

/api/Car/Apply

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

Для получения списка всех элементов справочника “Тип машины” необходимо выполнить запрос:

/api/catalogs/get?\$filter=type eq 'CarType'&\$expand=items(\$filter=isDeleted eq false)

3.4.2 Удаление машины

/api/Car/Delete

3.4.3 Получение списка машин

/api/Car/GetList

3.5 Справочник Прицепы

3.5.1 Добавление/редактирование прицепа

/api/Trailer/Apply

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

3.5.2 Удаление прицепа

/api/Trailer/Delete

3.5.3 Получение списка прицепов

/api/Trailer/GetList

3.6 Справочник Контрагенты

3.6.1 Добавление/редактирование контрагента

/api/CargoOwnerDictionary/Apply

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

3.6.2 Удаление контрагента

/api/CargoOwnerDictionary/Delete

3.6.3 Получение списка контрагентов

/api/CargoOwnerDictionary/GetList

3.7 Заявки

3.7.1 Получение адреса

/api/Map/GetAddresses

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

3.7.2 Создание/редактирование заявки

/api/TruckingBids/Apply

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

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](#)

3.7.3 Обновление заявки (только определенные поля)

/api/TruckingBids/Patch

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

3.7.4 Запуск заявки в работу

/api/TruckingBids/SetStatus

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

3.7.5 Получение списка заявок

/api/bids/GetListForExternal

3.7.6 Удаление заявки

/api/Bids/Delete

Удалять заявку можно только в статусе “Черновик”.

3.7.7 Отмена заявки

/api/Bids/Cancel

3.8 Трекеры

3.8.1 Привязка трекера к машине.

/api/Trackers/Attach

Для привязки необходимо указывать номер id трекера и машины.

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

/api/Trackers/Get

3.9 Сотрудники

3.9.1 Получение списка сотрудников организации

/api/Employees/GetNamesList

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.

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 обновятся из 1С.

Если выбран Cargo.Run приоритетной стороной, то при конфликте данные в 1С обновляются 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.