**Описание API**

**«Q-YMS»**

Версия 1.0

**Содержание**

Содержание

[1. Общая информация 3](#_Toc109403542)

[2. Авторизация 3](#_Toc109403543)

[3. Публичный API системы 4](#_Toc109403544)

[3.1 Пользователи системы 4](#_Toc109403545)

[3.2 Стойки системы 6](#_Toc109403546)

[3.3 Менеджер стоек 8](#_Toc109403547)

[3.4 Зоны проезда 10](#_Toc109403548)

[3.5 Доступы проездов 11](#_Toc109403549)

[3.6 Сессии проездов 13](#_Toc109403550)

[3.7 Работа с API плагинов 15](#_Toc109403551)

[3.7.1 API плагина KURSK\_AGRO 15](#_Toc109403552)

# Общая информация

Доступ к серверу и данным хранящимся в системе реализован в виде RESTfull архитектуры через наборы реализованных API.

CRUD – операции с данными CREATE/READ/UPDATE/DELETE

Сервер имеет следующие лимиты и обработчики по умолчанию:

* **Throttle** – 2000 параллельных запросов/сек
* **Read HeaderTimeout**: 5 секунд;
* **Idle Timeout**: 30 секунд;
* **Allowed Methods**: "GET", "POST", "PUT", "DELETE", "OPTIONS";
* **Allowed Headers**: "Access-Control-Allow-Headers",

"Access-Control-Allow-Origin", "Accept", "Authorization",

"Content-Type", "X-XSRF-Token", "X-JWT";

* **Exposed Headers**: "Authorization";

Стандартные обработчики:

* /ping – обработчик позволяющий определить работоспособность сервера. Отвечает pong

# Авторизация

Для авторизации используется следующий API URL:

http(s)://{server-host}:{port}/quath

Для данного API доступны следующие обработчики:

* /quath/login – выполняет авторизацию пользователя и выдает JWT токен

Метод **GET:** /quath/local/login?user=*user\_name*&passwd=*user\_passwd*

Метод **POST:** /quath/login

Body (JSON):

{

“user”:”user\_name”,

“passwd”:”user\_passwd”

**}**

* /quath/user – возвращает данные текущего пользователя на основе JWT

В ответ будут выданы cookie со значением JWT-токена.

Также авторизация запросов возможна на основе механизма BasicAuth.

# Публичный API системы

Все запросы выполняются с указанием версии API и имеют следующий вид базового URL:

http(s)://{server-host}:{port}/api/v1/

Схема протокола (HTTP или HTTP**S**) определяется настройками сервера.

❗️ - доступ к ресурсам помеченным знаком 🅰️ требует авторизации или JWT-токена. Ресурсы помеченные знаком 🆎 требуют доступа с ролью администратора.

* /info/\* – возвращает информацию о системе

{

"error":false,

"message":"OK, Q-YMS Version: 0.28.0",

"id":0,

"data":"0.28.0"

}

# Пользователи системы

Данный API позволяет выполнять CRUD операции с пользователями системы

Корневой URL ресурса:

http(s)://{server-host}:{port}/api/v1

* 🆎 GET **/users/{id|login}/\*** – возвращает информацию о пользователе по ID или Login

{

"id":"11111111",

"name":"",

"login":"admin",

"password":"",

"picture":"",

"role":"admin",

"billing\_id":"",

"admin":false,

"blocked":false

}

❗️ В целях безопасности пароль пользователя не может быть получен и не хранится в системе в открытом виде. Пароль может быть только изменен.

* 🆎 GET **/users -** возвращает список пользователей системы

[

{

"id":"1111111",

"name":"",

"login":"admin",

"password":"",

"picture":"",

"role":"admin",

"billing\_id":"",

"admin":false,

"blocked":false

},

{

"id":"2222222",

"name":"user",

"login":"user",

"password":"",

"picture":"",

"role":"user",

"billing\_id":"",

"admin":false,

"blocked":false

}

...

...

...

]

* 🆎 POST **/users –** создание нового пользователя в системе

{

"login":"super-user",

"name":"super user name",

"password":"super-secret-password",

"role":"user",

"blocked":false

}

**Ответ:**

{

"error":false,

"message":"user created",

"id":2224553101,

"data":null

}

* 🆎 PUT **/users/{id} –** обновление данных существующего пользователя в системе, где id соответствует id пользователя.

{

"id":"2222222",

"name":"user",

"login":"user",

"password":"",

"picture":"",

"role":"user",

"billing\_id":"",

"admin":false,

"blocked":false

}

* 🆎 DELETE **/users/{id} –** удаление существующего пользователя из системы, где id соответствует id пользователя.

**Ответ:**

{

"error":false,

"message":"user created",

"id":2222222,

"data":null

}

# Стойки системы

Данный API позволяет выполнять CRUD операции со стойками проезда в системе

Корневой URL ресурса:

http(s)://{server-host}:{port}/api/v1

* 🆎POST **/stands –** создание новой стойки в системе.

{

"attributes":{"attr\_name":"attr\_value"},

"uid":"5b4230cb-b9b9-4bbc-8a84-30e12d7d5bdd",

"name":"stand\_name",

"area\_id":2904780140,

"ip\_address":"192.168.1.110",

"type":2,

"device\_control\_protocol":"modbus"

}

**где type:**

**0 – въезд**

**1 – выезд**

**2 – въезд/выезд**

**3 – склад**

**uid – если значение при создании не указанно, то оно будет сгенерировано автоматически**

**Ответ:**

{

"id":3619506635,

"uid":"5b4230cb-b9b9-4bbc-8a84-30e12d7d5bdd",

"name":" stand\_name",

"ip\_address":"192.168.1.110",

"area\_id":2904780140,"type":2,

"state":0,

"location":{"latitude":0,"longitude":0},

"disabled":false,

"barrier\_disabled":false,

"passed\_is\_correct":false,

"sensors":null,

"warning\_timeout":0,

"warning\_message":"",

"is\_detecting":false,

"vehicle":{"plate":"","rfid":""},

"open\_manual":false,

"attributes":{"attr\_name":"attr\_value"},

"device\_control\_protocol":"modbus"

}

* 🅰️GET **/stands/{id|uid} –** получение информации о стойке в системе, где id соответствует **id** стойки или её **uid**.

**Ответ:**

{

"id":3619506635,

"uid":"5b4230cb-b9b9-4bbc-8a84-30e12d7d5bdd",

"name":" stand\_name",

"ip\_address":"192.168.1.110",

"area\_id":2904780140,"type":2,

"state":0,

"location":{"latitude":0,"longitude":0},

"disabled":false,

"barrier\_disabled":false,

"passed\_is\_correct":false,

"sensors":null,

"warning\_timeout":0,

"warning\_message":"",

"is\_detecting":false,

"vehicle":{"plate":"","rfid":""},

"open\_manual":false,

"attributes":{"attr\_name":"attr\_value"},

"device\_control\_protocol":"modbus"

}

* 🅰️PUT **/stands/{id} –** обновление информации по стойке в системе, где id соответствует **id** стойки.

{

"attributes":{"attr\_name":"attr\_value"},

"uid":"5b4230cb-b9b9-4bbc-8a84-30e12d7d5bdd",

"name":"stand\_name",

"area\_id":2904780140,

"ip\_address":"192.168.1.110",

"type":2,

"device\_control\_protocol":"modbus"

}

**где type:**

**0 – въезд**

**1 – выезд**

**2 – въезд/выезд**

**3 – склад**

* 🆎DELETE **/stands/{id} –** удаление стойки из системы, где id соответствует **id** стойки.

**Ответ:**

{

"error":false,

"message":"",

"id":3619506635,

"data":null

}

* 🅰️GET **/stands –** получение списка всех стоек системы.

Возвращает JSON массив объектов стойки.

* 🅰️GET **/stands/barrier/{uid} –** управление положением стрелы шлагбаума, где **uid** соответствует конкретной стойке.

{

"position":"**close|open**",

"uid":"5b4230cb-b9b9-4bbc-8a84-30e12d7d5bdd"

}

* 🅰️GET **/stands/out –** получение списка всех стоек системы имеющих тип «ВЫЕЗД» с текущими значениями состояний и параметров датчиков.

Возвращает JSON массив объектов стойки.

# Менеджер стоек

Данный API позволяет выполнять CRUD операции к менеджеру стоек, который обрабатывает текущие значения состояний, параметров и производит их обновление.

Корневой URL ресурса:

http(s)://{server-host}:{port}/api/v1

* 🅰️GET **/manager –** возвращает список стоек обслуживаемых менеджером системы (отколоченные или вновь добавленные стойки, до перезагрузки сервера, не обслуживаются менеджером).

Возвращает JSON массив объектов с данными стойки проезда.

* 🅰️POST **/manager/plc/{uid}?cmd={command\_name} –** отправляет команду на PLC контроллер, , где **uid** соответствует конкретной стойке

Body Данные в формате JSON для конкретного контроллера.

**Данные Modbus:**

{

"register\_number": 1, // номер регистра

"register\_value": 0, // значение регистра

"delay": 2 // значение в секунда для команды триггера

}

**Команды Modbus:**

"readRegister" – чтение значение регистра

"writeRegister" – запись значения в регистр

"triggerRegister" – переключение регистра с одного значения другое

и обратно

* 🅰️POST **/manager/{uid} – отправляется** запрос на проверку свободна ли зона.

Если запрос возвращает код ответа отличный от **200**, зона в которой находится стойка переполнена и въезд в неё запрещен.

* 🅰️PUT **/manager/{uid} – отправляется** запрос на обновление состояний стойки и её параметров.

Для обновления текущих параметров и состояний необходимо отправить JSON объект содержащий следующие поля:

{

"is\_detecting":false,

"vehicle":{"plate":"","rfid":""},

"open\_manual":false,

"sensors":null,

"attributes":{"attr\_name":"attr\_value"}

}

**is\_detecting** – признак того, что авто обнаружено около стойки посредством декторов (считыватель UHF или камера ГРНЗ)

**vehicle** – данные транспортного средства типа JSON объект содержащий следующие поля:

**plate** – номер ГРНЗ

**rfid** – идентификатор метки RFID

**open\_manual** – признак того, что шлагбаум открыт вручную (например, через кнопки на панели мониторинга)

**sensors** – данные состояний датчиков проезда и стойки. Содержат объект типа JSON:

sensors:{

in\_sensor: false, // датчик перед стойкой шлагбаума

out\_sensor: true, // датчик после стойки шлагбаума

barrier\_opened:true, // положение стрелы шлагбаума, поднят

barrier\_closed:false, // положение стрелы шлагбаума, опущен

traffic\_light:true // состояние светофора false-красный

}

**attributes** – произвольные атрибуты, которые могут быт добавлены к данным стойки, типа данных

"attributes":{

"attr\_name1":"attr\_value1",

"attr\_name2: false

…

}

**Ответ:** HTTP Status – 200 (OK)

* 🅰️GET **/manager/{uid} – возвращает данные стойки по UID**

Возвращает JSON объект данных стойки проезда.

# Зоны проезда

Данный API позволяет выполнять CRUD операции с зонами проезда системы

Корневой URL ресурса:

http(s)://{server-host}:{port}/api/v1

* 🆎POST **/areas –** добавление новой зоны в систему.

{

"id": 1299336677

"uid": "5b4230cb-b9b9-4bbc-8a84-30e12d7d5bdd",

"name":"area\_name",

"capacity":0 // 0 – неограниченно,

"check\_point":2,

"description":"some descriptions"

}

**Ответ:**

{

"id": 1299336677

"uid": "5b4230cb-b9b9-4bbc-8a84-30e12d7d5bdd",

"name":"area\_name",

"capacity":0 // 0 – неограниченно,

"check\_point":2,

"description":"some descriptions"

}

🅰️GET **/areas{id|uid} –** получение информации зоне в системе, где id соответствует **id** зоны или её **uid**.

**Ответ:**

{

"id": 1299336677

"uid": "5b4230cb-b9b9-4bbc-8a84-30e12d7d5bdd",

"name":"area\_name",

"capacity":0 // 0 – неограниченно,

"check\_point":2,

"description":"some descriptions"

}

* 🅰️GET **/areas –** получение списка всех проездов системы.

Возвращает JSON массив объектов зон проездов.

* 🅰️PUT **/areas/{id} –** обновление информации по зоне проезда в системе, где id соответствует **id** стойки.

{

"id": 1299336677

"uid": "5b4230cb-b9b9-4bbc-8a84-30e12d7d5bdd",

"name":"area\_name",

"capacity":0 // 0 – неограниченно,

"check\_point":2,

"description":"some descriptions"

}

**Ответ:**

{

"id": 1299336677

"uid": "5b4230cb-b9b9-4bbc-8a84-30e12d7d5bdd",

"name":"area\_name",

"capacity":0 // 0 – неограниченно,

"check\_point":2,

"description":"some descriptions"

}

* 🆎DELETE **/areas/{id} –** удаление зоны из системы, где id соответствует **id** стойки.

**Ответ:**

{

"error": false,

"message": "",

"id": 1299336677,

"data": null

}

# Доступы проездов

Данный API позволяет выполнять CRUD операции с доступами проезда в системе

Корневой URL ресурса:

http(s)://{server-host}:{port}/api/v1

* 🆎POST **/access –** добавление доступа.

{

"name":"АвтоИванова",

"plate\_number":"В112АР123",

"entry\_list":[2577719286,1742944929], // список ID стоек проезда

"description":"сотрудник отдела продаж",

"attributes":{}

}

**Ответ:**

{

“Id”: 749548260,

"name":"АвтоИванова",

"plate\_number":"В112АР123",

"entry\_list":[2577719286,1742944929],

"description":"сотрудник отдела продаж",

"attributes":{}

}

🆎GET **/access/{id|uid} –** получение информации по записи доступа проезда в системе, где id соответствует **id** зоны или её **uid**.

**Ответ:**

{

“Id”: 749548260,

"name":"АвтоИванова",

"plate\_number":"В112АР123",

"entry\_list":[2577719286,1742944929],

"description":"сотрудник отдела продаж",

"attributes":{}

}

* 🆎GET **/access –** получение списка всех доступов проезда системы.

Возвращает JSON массив объектов списка доступа.

* 🆎PUT **/access /{id} –** обновление информации по зоне проезда в системе, где id соответствует **id** стойки.

{

“Id”: 749548260,

"name":"АвтоПетрова",

"plate\_number":"В112АР123",

"entry\_list":[2577719286,1742944929],

"description":"сотрудник отдела продаж",

"attributes":{}

}

**Ответ:**

{

“Id”: 749548260,

"name":"АвтоПетрова",

"plate\_number":"В112АР123",

"entry\_list":[2577719286,1742944929],

"description":"сотрудник отдела продаж",

"attributes":{}

}

* 🆎DELETE **/access/{id} –** удаление доступа из системы, где **id** соответствует **id** доступа.

**Ответ:**

{

"error": false,

"message": "",

"id": 749548260,

"data": null

}

# Сессии проездов

Данный API позволяет выполнять CRUD операции с сессиями проезда в системе

Корневой URL ресурса:

http(s)://{server-host}:{port}/api/v1

* 🅰️POST **/sessions/{uid} –** добавление сессии проезда.

{

"uid":<stand\_uid>,

"passed\_is\_correct": <true|false>

}

**Ответ (объект session):**

{

"id": "f6dd03c4-ec4e-499e-a6b1-eb8de144f72c",

"stand\_id": 1742944929,

"area\_id": 1633299711,

"rfid": "",

"plate\_number": "b078hy193",

"is\_open\_manual": false,

"is\_paid": false,

"payment\_id": "",

"payment\_required": false,

"attributes": {

"camera\_channel": "eYT5ULwU",

"entry": "1",

"manual\_open": true,

"plate": "b078hy193",

"rfid": "",

"source": "ПрибылАвторизован"

},

"vehicle": {

"plate": "b078hy193",

"rfid": ""

},

"status": 2001,

"start\_time": 1652466531,

"end\_time": 1652466593,

"total\_time": 62

}

🅰️GET **/sessions/{id} –** получение данных сессии, где id соответствует **id** сессии.

**Ответ:**

{

"id": "f6dd03c4-ec4e-499e-a6b1-eb8de144f72c",

"stand\_id": 1742944929,

"area\_id": 1633299711,

"rfid": "",

"plate\_number": "b078hy193",

"is\_open\_manual": false,

"is\_paid": false,

"payment\_id": "",

"payment\_required": false,

"attributes": {

"camera\_channel": "eYT5ULwU",

"entry": "1",

"manual\_open": true,

"plate": "b078hy193",

"rfid": "",

"source": "ПрибылАвторизован"

},

"vehicle": {

"plate": "b078hy193",

"rfid": ""

},

"status": 2001,

"start\_time": 1652466531,

"end\_time": 1652466593,

"total\_time": 62

}

* 🅰️GET **/session –** получение списка всех доступов проезда системы.

Возвращает JSON массив объектов сессиий ([]session).

* 🅰️PUT **/session /{id} –** обновление данных сессии, где id соответствует **id** сессии. В теле может присутствовать как одно поле, так и несколько полей объекта session

{

"rfid": "",

"plate\_number": "b078hy193",

"is\_open\_manual": false,

"attributes": {

"camera\_channel": "eYT5ULwU"

}

}

**Ответ: -** Обновленный объект Session

* 🆎DELETE **/session/{id} –** удаление сессии из системы, где **id** соответствует **id** существующей сессии.

**Ответ:**

{

"error": false,

"message": "",

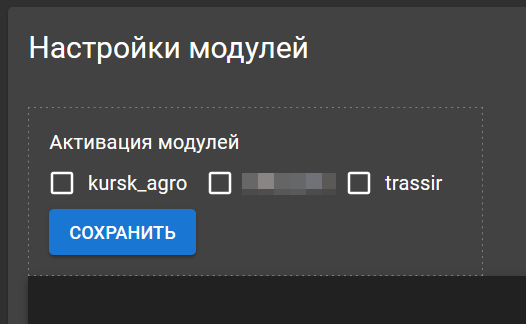
"id": 749548260,

"data": null

}

# Работа с API плагинов

Перед началом работы с функционалом плагинов убедиться, что соответствующий плагин включен в настройках. Добавление функционала плагинов осуществляется путем добавления плагина в основной файл конфигурации (список поддерживаемых плагинов запрашивается индивидуально), после чего его можно будет активировать в разделе настроек модулей WEB-интерфейса:



"plugins": [  
 "plugin\_name",  
 "trassir"  
]

После активации плагина можно переходить к его настройке.

Все плагины реализуют единый интерфейс PluginInterface, в котором реализован универсальный метод обработки команд.

CommandExecutor(cmd string, data interface{}) (result interface{}, err error)

**Корневой URL ресурса плагинов:**

http(s)://{server-host}:{port}/api/v1/plugin

# API плагина KURSK\_AGRO

Для работы с данным плагином необходимо использовать поддерживаемые команды плагина:

* **check\_vehicle** – выполняет проверку и авторизацию транспортного средства на въезд для конкретной точки проезда. В качестве данных для проверки используются входящие данные:

{

"uid": <uid\_проезда>,

"plate\_number": <ГРНЗ ТС>,

"rfid": <rfid\_id>

}

Если проезда разрешен, то будет отправлен HTTP ответ c кодом *200*. Ответ будет содержать JSON объект вида:

{  
"#ns": "https://sodrugestvo.com/TransportManagement.... ",  
"#type": "TransportRegistrationMessage",  
"#value": {  
 "ID": "ecfd7149-ad82-11eb-a838-000d3aaa9d83",  
 "Entry": 4,  
 "RFID": "RFID",  
 "Manual": true  
 }  
}

В случае ошибки будет возвращен HTTP код ошибки (4XX-500)

🅰️POST **/kursk\_agro?cmd=check\_vehicle**

* **confirm\_passed** – фиксирует завершение проезда и выставляет советующий признак корректности проезда. В качестве данных тела запроса используется объект данных сессии для текущего проезда (получение данных сессии можно выполнить через API сессий для текущего проезда):

{

"id": "f6dd03c4-ec4e-499e-a6b1-eb8de144f72c",

"stand\_id": 1742944929,

"area\_id": 1633299711,

"rfid": "",

"plate\_number": "b078hy193",

"is\_open\_manual": false,

"is\_paid": false,

"payment\_id": "",

"payment\_required": false,

"attributes": {

"camera\_channel": "eYT5ULwU",

"entry": "1",

"manual\_open": true,

"plate": "b078hy193",

"rfid": "",

"source": "ПрибылАвторизован"

},

"vehicle": {

"plate": "b078hy193",

"rfid": ""

},

"status": 2001,

"start\_time": 1652466531,

"end\_time": 1652466593,

"total\_time": 62

}

Если сессия существует и данные обработаны корректно, то будет отправлен HTTP ответ c кодом *200*. Ответ будет содержать JSON объект вида:

{

                "#ns": "https://sodrugestvo.com/.....",

                "#type": "DepartureTransportMessage",

                "#value": {

             "ID": "ecfd7149-ad82-11eb-a838-000d3aaa9d83

                 "Entry": 4,

                 "RFID": "RFID",

                "DateSelect": "2021-05-28T09:00:00",

                "DateOut": "2021-05-28T09:00:00",

                "Result": true

                }

}

🅰️POST **/kursk\_agro?cmd=confirm\_passed**

* **route\_data\_rfid** – выполняет запрос данных рейса на основании привязанного RFID идентификатора. В качестве данных тела запроса используется RFID идентификатор:

{

"rfid":"E1000010100101E2"

}

Если данные маршрута существуют и данные обработаны корректно, то будет отправлен HTTP ответ c кодом *200*. Ответ будет содержать JSON объект вида:

{

"#ns": "https://sodrugestvo.com/TransportManagement/WeighingShipment/1.2.1.1",

"#type": "TransportEntryRequestMessage",

"#value":{

"ID": "d4a181e2-c2d5-11eb-a839-000d3aaa9d83",

"Entry": 1, <integer>

"RFID": " E1000010100101E2",

"Cargo":{"CargoName": "Phasol", "Brutto": 0, "Netto": 0, " TareWeight": 0},

"CarNum": "",

"Driver": "Чижолв ВВ",

"Cancel": false

}

}

В случае ошибки будет возвращен HTTP код ошибки (4XX-500)

🅰️POST **/kursk\_agro?cmd=route\_data\_rfid**

* **fetch\_weight** – выполняет запрос данных веса через API драйвера весовых терминалов NAIS. В качестве данных тела запроса используется RFID и номер проезда:

{

"rfid":"E1000010100101E2",

"entry":"1"

}

Если данные обработаны корректно, то будет отправлен HTTP ответ c кодом *200*. Ответ будет содержать JSON объект вида:

{

"id": " d4a181e2-c2d5-11eb-a839-000d3aaa9d83",

"entry": 1,

"weight": 1024.48

}

В случае ошибки будет возвращен HTTP код ошибки (4XX-500)

🅰️ POST **/kursk\_agro?cmd=send\_weight\_result**

* **send\_weight\_result** – отправляет результаты взвешивания в очередь RMQ. В качестве данных тела запроса используется данные веса:

{

"id":" d4a181e2-c2d5-11eb-a839-000d3aaa9d83",

"entry":1,

"rfid":"00ff00ff00ff",

"weight":34.4253,

"date":1618079262,

"accept":true

}

Если данные веса обработаны корректно, то будет отправлен HTTP ответ c кодом *200*. Ответ будет содержать JSON объект вида:

{

"#ns": "https://sodrugestvo.com/TransportManagement/WeighingShipment/1.2.1.1",

"#type": "WeighResultsMessage",

"#value":{

"ID": "0325b75a-99d0-11eb-a8b3-0242ac130003",

"Entry": 1,

"RFID": "00ff00ff00ff",

"DateEnd": "2021-07-01T19:59:44.92",

"Weight": 34.4253,

"accept": false

}

}

В случае ошибки будет возвращен HTTP код ошибки (4XX-500)