Подключение к Mobile SMARTS через REST API

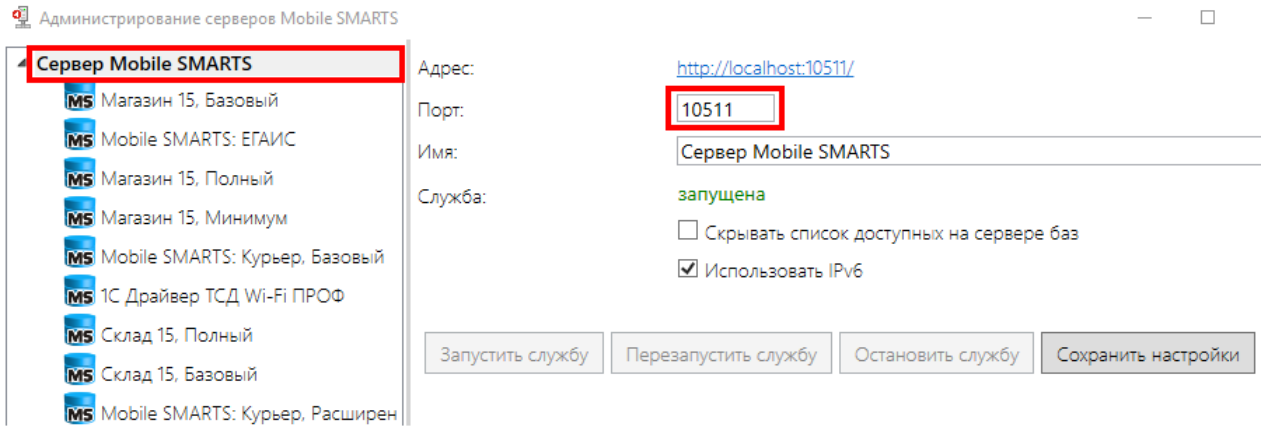
Последние изменения: 2024-03-26

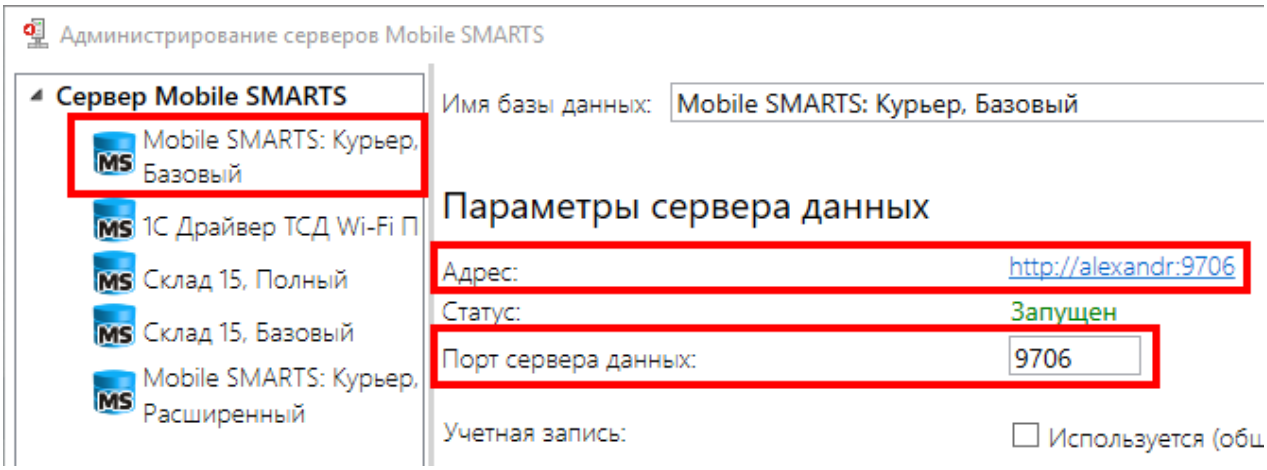
Точкой входа для доступа к API в нашей системе является url вида:

http(s)://{имя сервера}:{порт базы}/api/v1

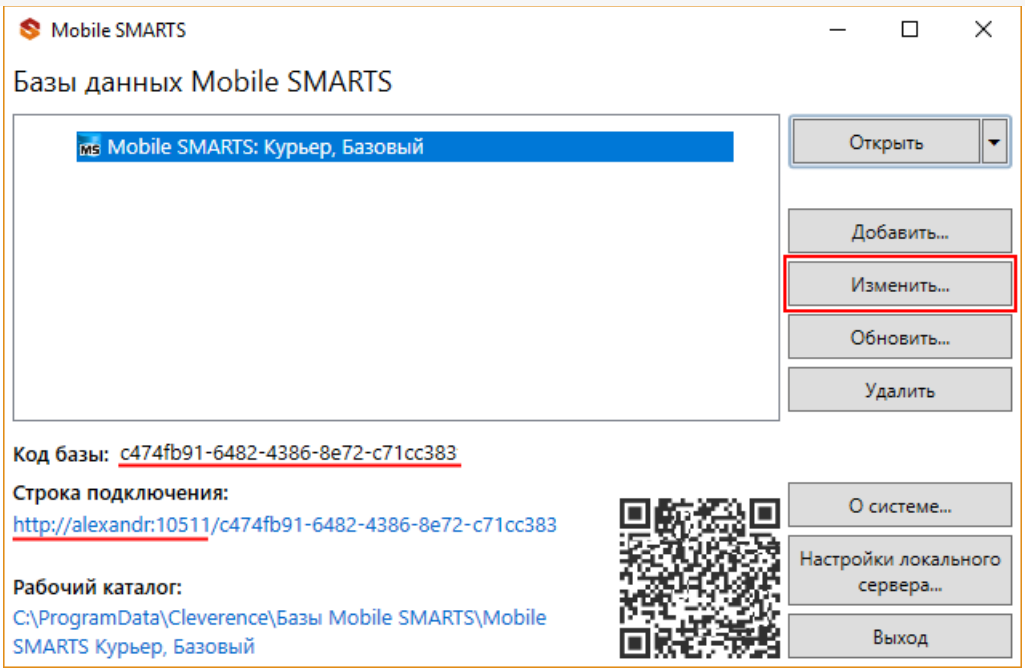
По нему можно получить список методов апи.

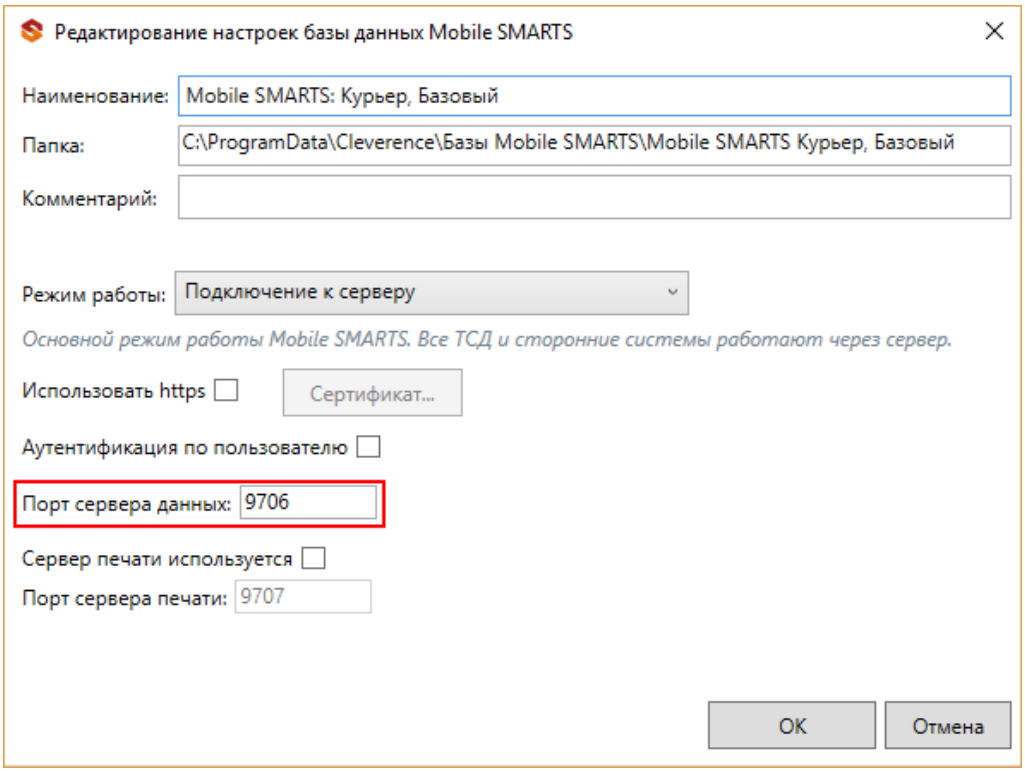
Чтобы узнать порт сервера, порт базы или идентификатор базы, надо воспользоваться приложением для администрирования сервера Mobile SMARTS.





Или менеджером баз данных Mobile SMARTS.





Для приведенных скриншотов url получаются такими:

http(s)://localhost:10511/c474fb91-6482-4386-8e72-c71cc383/api/v1

или

http(s)://localhost:9706/api/v1

интеграция, REST API

Не нашли что искали?

Задать вопрос в техническую поддержку

Проверка запросов через Swagger

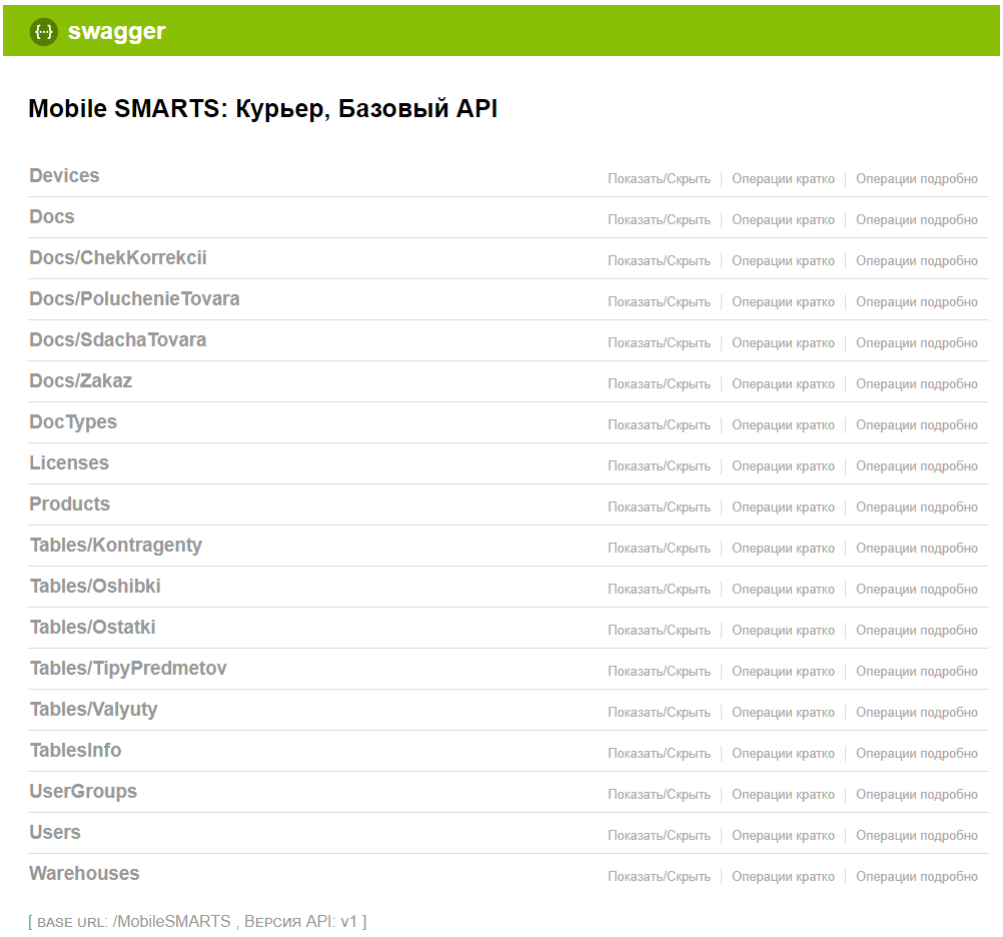
Последние изменения: 2024-03-26

> Примеры Swagger для Mobile SMARTS:
> Swagger для «Mobile SMARTS: Магазин 15»
> Swagger для «Mobile SMARTS: Курьер»

В нашей системе также предусмотрена возможность просматривать структуру API и выполнять простые запросы, используя Swagger. Для доступа к нему необходимо в браузере зайти по адресу:

http://xxx.xxx.xxx.xxx:9000/MobileSMARTS/swagger/ui/index

По этому адресу открывается страница с описанием всех методов.



Каждая группа в этом списке разворачивается и отображает все методы этой группы.

Mobile SMARTS: Курьер, Базовый API

Devices Показать/Скрыть | Операции кратко | Операции подробно

GET	/api/v1/Devices	Список устройств
POST	/api/v1/Devices	Добавить/отредактировать устройство
DELETE	/api/v1/Devices('{deviceId}')	Удалить устройство
GET	/api/v1/Devices('{deviceId}')	Получить устройство по идентификатору
PATCH	/api/v1/Devices('{deviceId}')	Изменить устройство
PUT	/api/v1/Devices('{deviceId}')	Добавить/отредактировать устройство по известному идентификатору

Docs Показать/Скрыть | Операции кратко | Операции подробно

Docs/ChekKorrekcii Показать/Скрыть | Операции кратко | Операции подробно

В каждом методе можно просматривать все входящие и исходящие параметры. Также можно просмотреть описание сущностей, которые отправляются/получаются при использовании этого метода.

Devices Показать/Скрыть | Операции кратко | Операции подробно

GET /api/v1/Devices — Список устройств

Пример ответа (Статус 200)

OK

Описание | Пример


```
{
  "@odata.context": "string",
  "value": [
    {
      "appInstanceId": "string",
      "deviceId": "string",
      "batteryStatus": "string",
      "lastInfoTime": "2018-03-20T06:49:48.982Z",
      "userId": "string",
      "warehouseId": "string",
```


Content Type ответа application/json

Параметры

Параметр	Значение	Описание	Тип параметра	Тип данных
$expand		Expands related entities inline.	query	string
$filter		Filters the results, based on a Boolean condition.	query	string
$select		Selects which properties to include in the response.	query	string
$orderby		Sorts the results.	query	string
$top		Returns only the first n results.	query	integer
$skip		Skips the first n results.	query	integer
$count		Includes a count of the matching results in the response.	query	boolean

Попробовать!

Также можно получить json схемы всех методов апи и по нему генерировать модели в своем приложении.

http://localhost:{порт базы}/swagger/docs/v1 - получение json сваггера.

интеграция, REST API, Swagger

Не нашли что искали?

Задать вопрос в техническую поддержку

Авторизация в системе через REST API

Последние изменения: 2024-03-26

Если в системе включена авторизация, то для начала работы с API необходимо пройти авторизацию в системе.

Реализованы несколько вариантов авторизации:

- BASIC авторизация.
- Авторизация методом GET по адресу /api/session с получением token.
- Авторизация методом POST по адресу /api/session с получением token.

REST API работает только если при работе с ТСД используется сервер Mobile SMARTS. При прямом подключении ТСД к компьютеру через кабель/крэдл или при обмене с учетной системой через папку использовать REST API не получится.

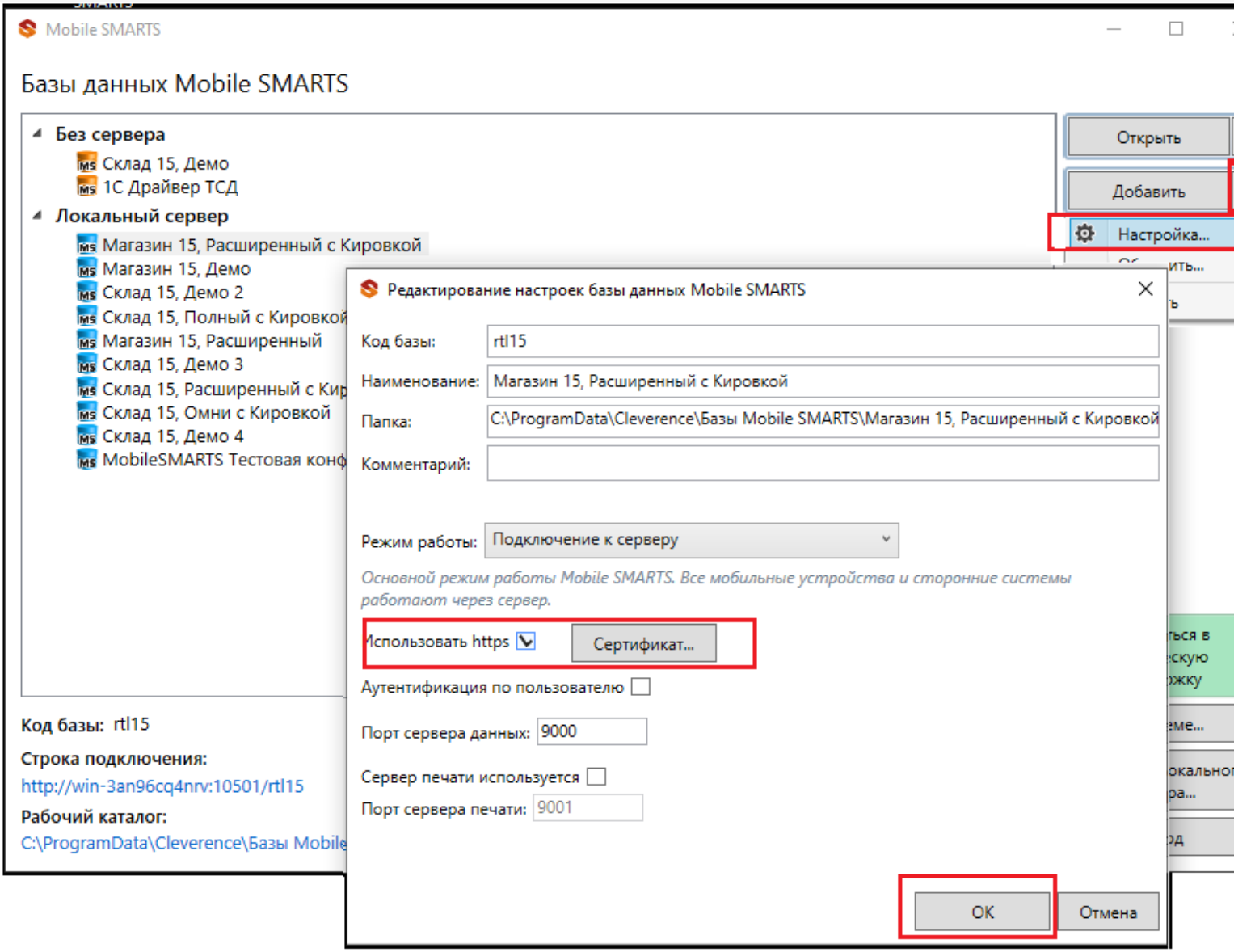
В базе Mobile SMARTS должна быть включена авторизация, тогда для выполнения HTTP-запросов необходимо будет каждый раз авторизовываться, есть 2 варианта авторизации:

- BASIC авторизация — в этом случае при каждом HTTP-запросе нужно будет отправлять логин и пароль;

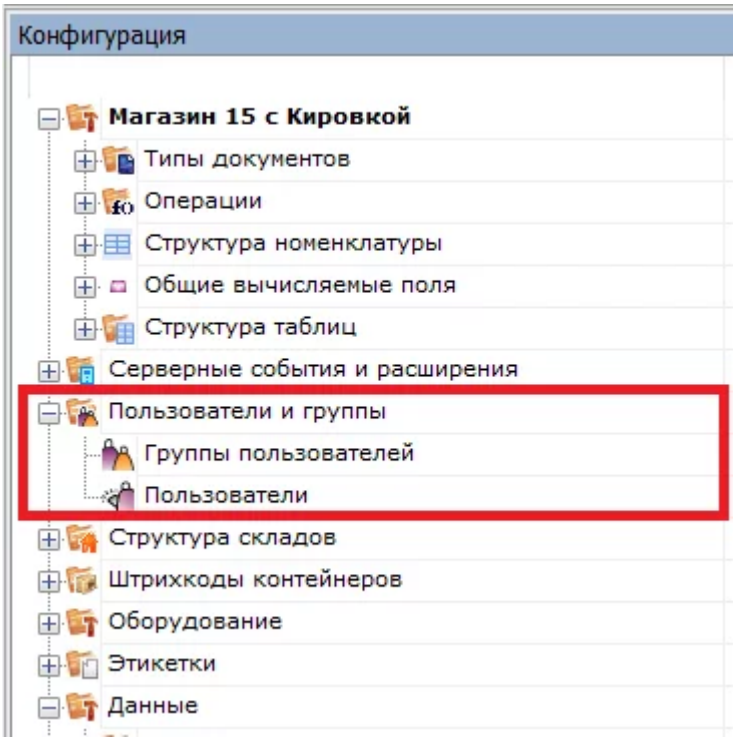
- авторизация с использованием токена — при первом HTTP-запросе передаются логин и пароль, а сервер MS возвращается токен — уникальный идентификатор сессии (access token), который при последующих HTTP-запросах можно будет использовать вместо передачи логина и пароля. Срок действия токена (т.е. сессии) ограничен, поэтому для обновления токена при первом запросе сервер Mobile SMARTS дополнительно возвращает «токен обновления» (refresh token) — он нужен для получения нового токена после истечения срока действия текущего токена.

Обратите внимание. Для защиты передаваемых данных рекомендуется использовать режим с включенной аутентификацией и доступом по https.

Для работы через протокол https необходимо указать это в настройках базы данных, для этого зайдите в менеджер базы, справа кнопки «Добавить» нажмите на выпадающий список и выберите пункт «Настройки», далее в окне «Редактирование настроек» поставьте галочку «Использовать http» и нажмите «ОК».



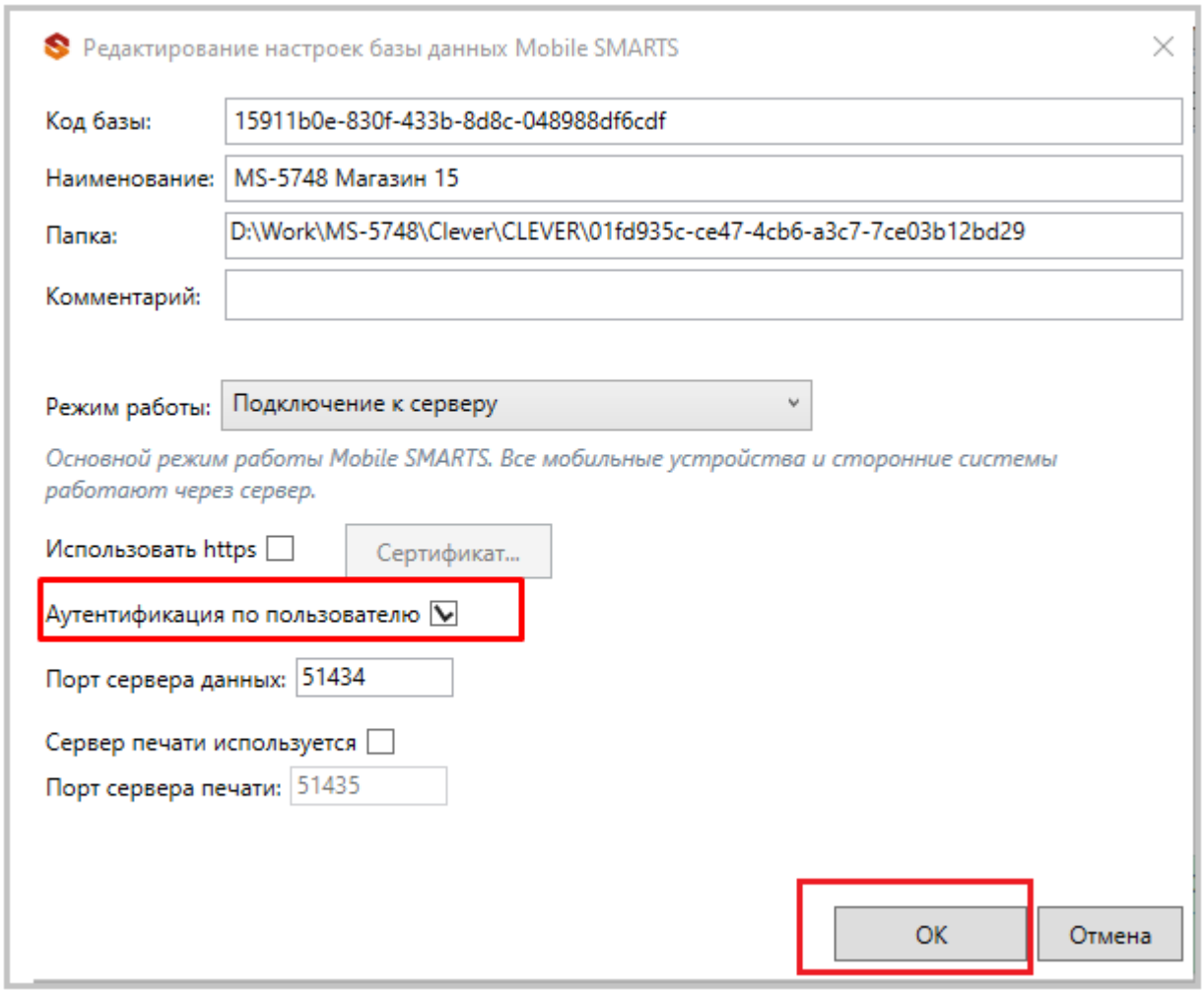
Для работы в режиме с включенной аутентификацией необходимо завести пользователей, задать им логины и пароли. В панели управления Mobile SMARTS узел «Пользователи и группы» содержит данные о пользователях и группах, в которых они состоят.



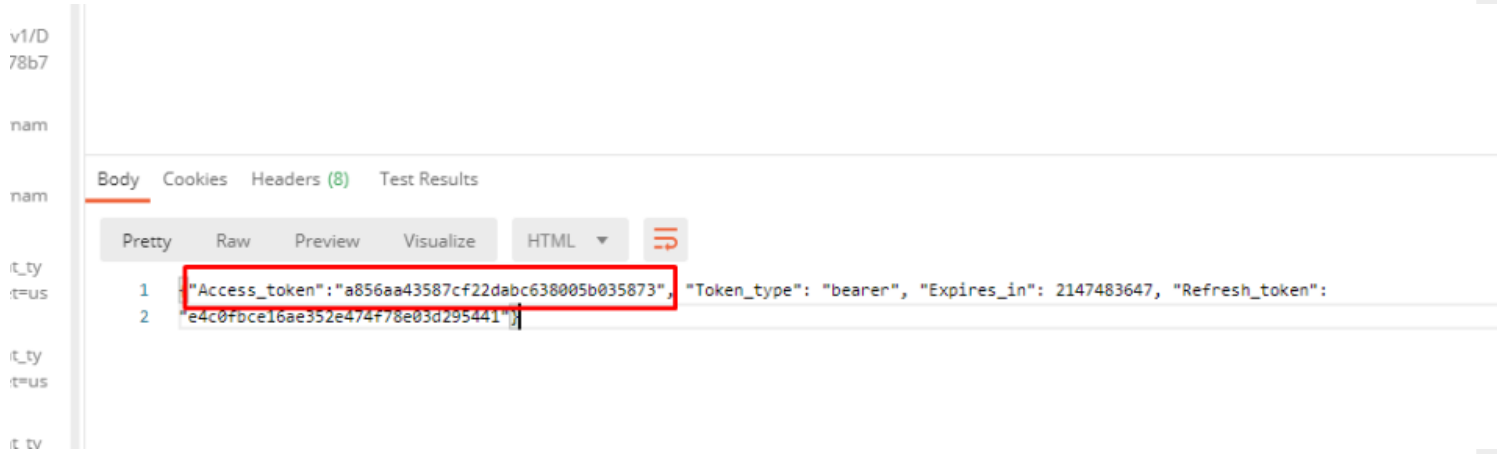
Инструкция по конфигурации настроек пользователей и групп пользователей, как определить роль пользователей и список типов документов, доступных для обработки пользователям такой группы описана в одноименной статье на сайте.

Пример запроса на авторизацию приложения методом POST

1. Для начала в настройках базы Mobile SMARTS включите авторизацию/аутентификацию по пользователю и нажмите на кнопку «ОК»



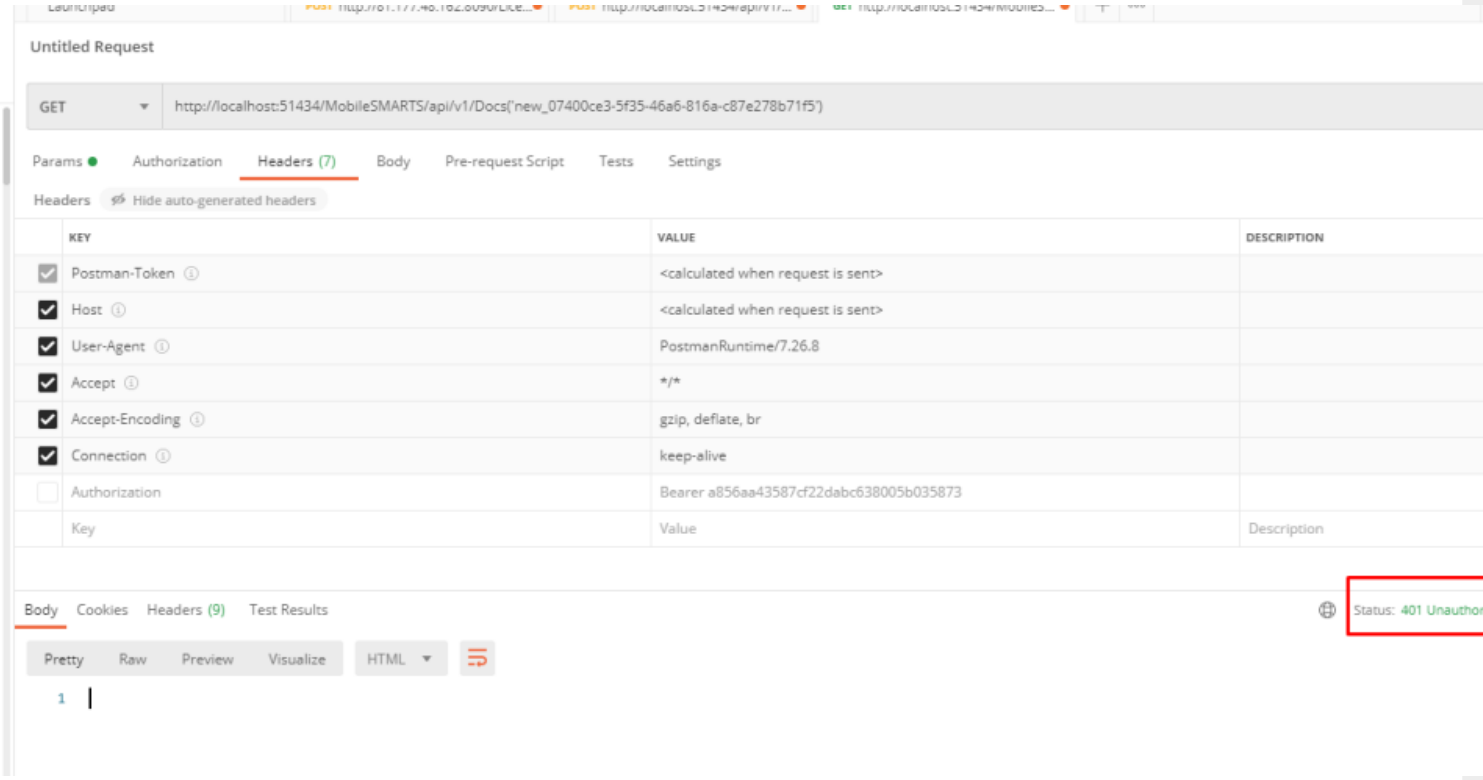
2. Пример запроса на авторизацию
http://localhost:51434/api/v1/session?username={username}&password={password}
в ответ приходит Access_token



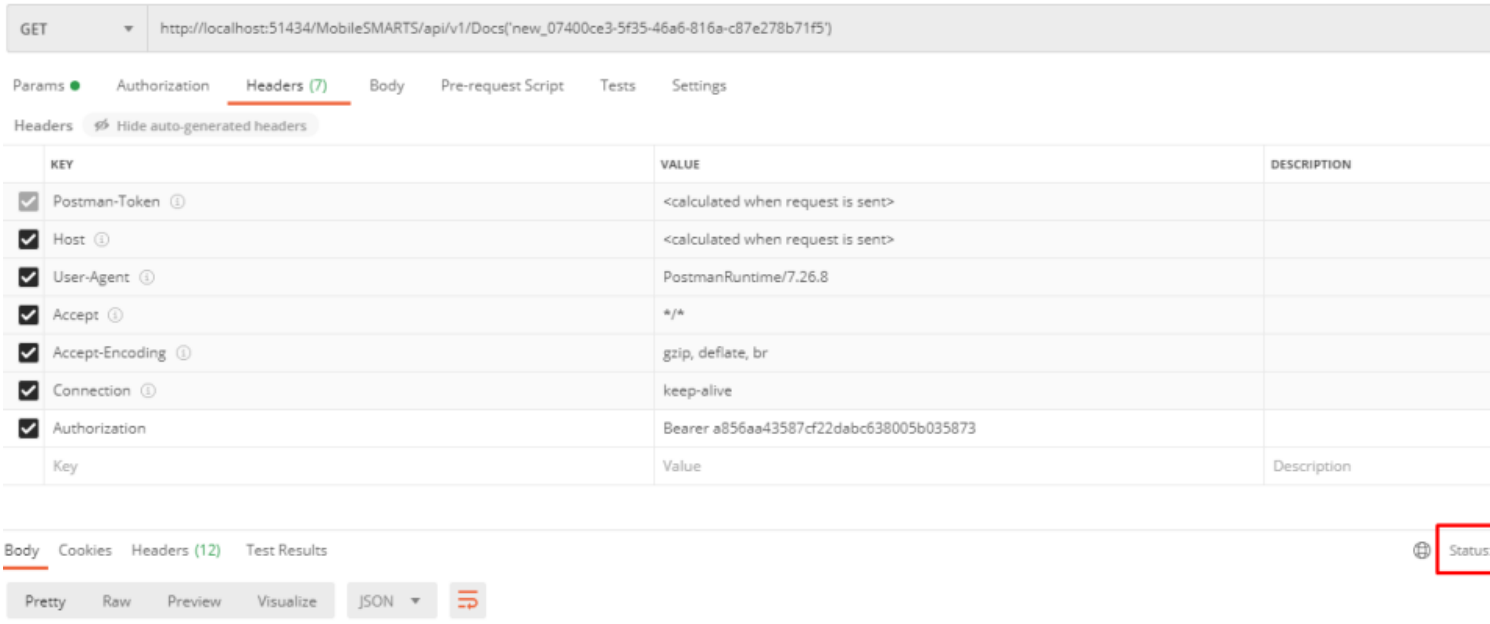
3. В последующих запросах в заголовке указываем

{Authorization: Bearer {Access_token}

Если не указать Authorization, в ответ приходит статус ошибки
401 — Unauthorized



4. При успешной авторизации приходит ответ и статус 202 — OK



BASIC авторизация

Данный вид авторизации чаще всего используется браузером для доступа к функциям API.

При использовании данного метода необходимо в заголовке каждого запроса указывать:

```
Authorization: Basic {login}:{password}
```

Допускается base64 при формировании строки {login}:{password}

Авторизация методом GET

Еще один способ авторизации — отправить GET запрос по адресу /api/v1/session, при этом в url запросе указать параметры login и password:

https://localhost:9000/api/v1/session?username=${Username}&password=${Password}

Ответ сервера:

```
{
Access_token:"123123123",
Token_type:"bearer",
Expires_in:86400,
Refresh_token:"321321321",
}
```

Авторизация методом POST

Авторизация по логину и паролю происходит путем отправки POST запроса на сервер, в результате которого возвращается access_token и refresh token в формате JSON.

Пример запроса:

POST /oauth/token HTTP/1.1
Host: mobilesmarts.ru/api/session
Content-Type: application/x-www-form-urlencoded

grant_type=password&client_id=123&client_secret=user&username=user@domain.ru&password=123456

Ответ сервера:

```
{
Access_token:"123123123",
Token_type:"bearer",
Expires_in:86400,
Refresh_token:"321321321",
}
```

Восстановление после окончания срока действия сессии

Восстановление после окончания срока действия сессии происходит путем отправки refresh_token на сервер, в результате приходит новый access_token и refresh_token

Пример:

POST /oauth/token HTTP/1.1

Host: mobilesmarts.ru/api/session
Content-Type: application/x-www-form-urlencoded

grant_type:refresh_token&client_id=123&client_secret=user&refresh_token=321321321

Ответ:

HTTP/1.1 200 OK
Content-Type: application/json

```
{
Access_token:"99999",
Token_type:"bearer",
Expires_in:86400,
Refresh_token:"789789789",
}
```

Вызов функций с использованием token

Для того чтобы обратиться к функциям (если не используется Basic авторизация), для которых необходима авторизация, необходимо в заголовке Authorization передавать токен:

Authorization: Bearer <token>

Иначе сервер вернет ошибку авторизации 401.

интеграция, REST API

Не нашли что искали?

Задать вопрос в техническую поддержку

Формат запросов в REST API

Последние изменения: 2024-03-26

Формат передачи входящих и исходящих данных — JSON, в кодировке UTF-8. Входящие параметры должны передаваться в теле POST запроса или в виде query-строки (?field=value) для GET запросов. Нужно учитывать, что GET запросы имеют ограничение на длину URL — 2048 символов.

В случае, если указан неверный адрес — возвращается ошибка 404.

В случае возникновения ошибки авторизации, возвращается ошибка 401.

Запросы строятся в формате OData, подробнее можно посмотреть по адресу

http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part2-url-conventions.html

Пагинация

Функции, возвращающие массив значений, поддерживают параметры для порционной загрузки, которые передаются в виде query-строки (?$top=1&$skip=10).

Пример:

https://localhost:9000/api/v1/Docs?$top=5&$skip=10

При выполнении данного запроса мы получим максимум 5 записей начиная с 11 индекса в списке.

Если в запросе на получение списка указать параметр count=true, в результате вернется дополнительно поле count, которое содержит общее число записей в списке на сервере.

Фильтрация

Для фильтрации данных в запросе можно указывать параметр $filter, например:

https://localhost:9000/api/v1/Docs?$filter=lastChangeDate gt 2017-10-06T17:41:10Z and documentTypeName eq 'Заказ'

При выполнении данного запросы мы получим список документов, дата изменения которых более указанной в запросе и тип документа — «Заказ».

Выборка только нужных свойств

Также можно указывать список полей, которые необходимо отобразить в результате, например:

https://localhost:9005/api/v1/Docs?$select=id,tables

Результатом выполнения данного запроса будет список документов, состоящих только из 2-х полей — id и tables.

Выборка записи по идентификатору

Для получения нужной записи необходимо составить запрос с указанием идентификатора этой записи, например:

https://localhost:9005/api/v1/Docs ('165')

Сортировка

Чтобы отсортировать результат необходимо указать параметр $orderby, например:

https://localhost:9005/api/v1/Docs?$orderby=id

В результате получим список документов, отсортированный по идентификатору.

Отображение подтаблиц и строк документов

Для документов реализован вывод записей строк документа и таблиц документа.

Пример:

Если в документе с идентификатором «1» существует таблица «ОплатыВозвраты», то получить строки таблицы можно отправив запрос:

https://localhost:9000/api/v1/Docs ('1')/ОплатыВозвраты

Для получения только строк документа с идентификатором «123» необходимо выполнить запрос

https://localhost:9000/api/v1/Docs ('123')/declaredItems это запрос плановой части документа

http://localhost:9000/api/v1/Docs ('123')?$expand=currentItems запрос фактической части документа

Развертывание сложных свойств

Чтобы отобразить все документы, включая его таблицы со строками, нужно сформировать запрос в таком виде:

https://localhost:9000/api/v1/Docs?$expand=tables ($expand=rows)

Поиск без учета регистра

https://localhost:9000/api/v1/Docs?$filter=Contains (tolower (name), 'фраза')

Запросы DELETE

Для запросов DELETE требуется параметр If-Match, нужно указывать *.

Добавление массива данных

В некоторых методах предусмотрено добавление массива

{ «value»:[{первый элемент}, {второй элемент}]}

интеграция, REST API

Не нашли что искали?

Задать вопрос в техническую поддержку

Примеры запросов в REST API

Последние изменения: 2024-03-26

Получение информации о базе

Для получения информации по текущей базе нужно выполнить запрос:

GET /api/v1/BaseInfo

Ответ:


```
{
"@odata.context": "http://172.19.0.30:9000/MobileSMARTS/api/v1/$metadata#BaseInfo",
"id": "rtl15",
"name": "Магазин 15, Базовый",
"folder": "C:ProgramDataCleverenceБазы Mobile SMARTSМагазин 15, Базовый",
"appId": "F42C7B5F-405C-4076-AE07-9348F189EE71",
"appName": "Магазин 15, Базовый",
"comment": null,
"allConnectionStrings": [
"https://172.19.0.30:10502/rtl15",
"Доступ к swagger:"
"https://172.19.0.30:10502/rtl15/swagger"
],
"appDescription": {
"appId": "F42C7B5F-405C-4076-AE07-9348F189EE71",
"appName": "Магазин 15, Базовый",
"desktopReqPayment": false,
"configId": "8DC2C8BA-77CE-46AD-93CB-6CFFCA7B96D7",
"serverModeSupported": true,
"onlineCallsSupported": false,
"batchModeSupported": true,
"appLink": "http://cleverence.ru/software/mobile-smarts/rtl15/",
"appSupportLink": "https://www.cleverence.ru/support/category:295/",
"aboutLicLink": "http://cleverence.ru/software/mobile-smarts/rtl15/#spec",
"appVersionsLink": "http://cleverence.ru/software/mobile-smarts/rtl15/#spec",
"comment": "Основная поставка Магазин 15",
"appVersion": "1.1.1.155",
"clientVersion": "3.0.0.100",
"panelVersion": "1.0",
"mainServerVersion": "1.0",
"appServerVersion": "3.0.0.2699",
"minPlatformVersion": "1.0",
"minPlatform35Version": "1.0"
},
"appInstanceSettings": {
"mode": "Server",
"hasServerAuth": false,
"serverSettings": {
```

```
"dataService": {
"enabled": true,
"port": 9000,
"useHttps": false,
"deviceAuth": false,
"passwordAuth": false
},
"printService": {
"enabled": false,
"port": 9001,
"useHttps": false,
"deviceAuth": false,
"passwordAuth": false
}
}
}
}
```


Группы пользователей

Работа с массивом
GET /api/v1/UserGroups


```
 {
"@odata.context": "string",
"value": [
{
"id": "string",
"name": "string",
"documentTypeNames": [
"string"
],
": true,
"role": 0,
"serverSideInventory": true,
"autorunDocumentTypeName": "string",
"onStartHandlerName": "string",
"onFinishHandlerName": "string"
}
]
 }
```


Работа с записями по идентификатору

POST /api/v1/UserGroups ('{key}')

PUT /api/v1/UserGroups ('{key}')

PATCH /api/v1/UserGroups ('{key}')

DELETE /api/v1/UserGroups ('{key}')

UserGroup {

id (string, *optional*): Id пользователя ,
name (string, *optional*): Уникальное наименование группы ,
documentTypeNames (Array[string], *optional*, read only),
batchMode (boolean, *optional*): Свойство, позволяющее задать тип обмена данными с сервером. Если true - обмен производится вручную, по запросу пользователем. Иначе, обмен будет производиться периодически, с интервалом, заданным в настройках клиентского приложения ,
role (integer, *optional*): Пользовательская роль = ['0', '1', '2', '3'],
serverSideInventory (boolean, *optional*): Указывает источник номенклатурного справочника. Если true - позиции номенклатуры будут искаться на сервере(необходимо наличие постоянной связи с сервером), иначе - справочник номенклатуры будет загружаться на терминал при операции обмена данными и использоваться локально ,
autorunDocumentTypeName (string, *optional*): Тип документа (виртуальный), который будет открыт автоматически, сразу после запуске программы на ТСД ,
onStartHandlerName (string, *optional*): Имя операции, запускающейся при входе пользователя данной группы ,
onFinishHandlerName (string, *optional*): Имя операции, запускающейся при выходе пользователя данной группы

}

Получение списка пользователей группы

GET /api/v1/UserGroups ('{key}')/users

Пользователи

Работа с массивом

GET /api/v1/Users

Работа с записями по идентификатору

POST /api/v1/Users

PUT /api/v1/Users

PATCH /api/v1/Users

DELETE /api/v1/Users

User {

id (string, *optional*): Id пользователя ,

name (string, *optional*): Имя пользователя ,

password (string, *optional*): (write only) Пароль пользователя ,

description (string, *optional*): Описание пользователя ,

barcode (string, *optional*): (write only) Штрихкод пользователя ,

groupId (string, *optional*): Имя группы в которой состоит пользователь. Подробнее смотрите UserGroup ,

groupName (string, *optional*): Для совместимости со старыми обработками ,

warehouseIds (Array[string], *optional*, read only)

}

Ячейки

GET /api/v1/Cells — Список ячеек

Работа с записями по идентификатору

POST /api/v1/Cells — Добавить/редактировать ячейку

DELETE /api/v1/Cells ('{id}') — Удалить ячейку

GET /api/v1/Cells ('{id}') — Получить ячейку по идентификатору

PUT /api/v1/Cells ('{id}') — Добавь/редактировать ячейку по идентификатору

Обновление справочника со сбросом всех записей

POST /api/v1/Cells/BeginOverwrite — начинает процедуру пакетной выгрузки ячеек на сервер. Все позиции будут накапливаться и не будут доступны до вызова функции EndOverwrite.

POST /api/v1/Cells/EndOverwrite — завершает процедуру пакетной выгрузки номенклатуры. Только после вызова этой функции сервер завершит обработку переданных позиций номенклатуры и они попадут в справочник товаров. Старый справочник товаров при этом будет полностью очищен.

Обновление записей

POST /api/v1/Cells/BeginUpdate — начинает процедуру пакетного обновления ячеек на сервере. Все позиции будут накапливаться и не будут доступны до вызова функции EndUpdate.

POST /api/v1/Cells/EndUpdate — завершает процедуру пакетного обновления ячеек. Только после вызова этой функции сервер завершит обработку переданных позиций номенклатуры и они попадут в справочник товаров. Выгруженная номенклатура будет слита с существующим на сервере справочником.

Сброс обновления/ перезаписывания

POST /api/v1/Cells/ResetUpdate — сбрасывает процедуру пакетного обновления ячеек.

Номенклатура

Получение схемы

Получение списка всех полей номенклатуры:

https://localhost:9000/api/v1/ProductSchema?$expand=allfields

Получение полей номенклатуры:

https://localhost:9000/api/v1/ProductSchema?$expand=fields

Получение списка фиксированных полей:

https://localhost:9000/api/v1/ProductSchema?$expand=defaultFields

Работа с массивом

GET /api/v1/Products — список номенклатуры.

Работа с записями по идентификатору

POST /api/v1/Products — добавить/ редактировать номенклатуру.

DELETE /api/v1/Products ('{id}') — удалить номенклатуру.

GET /api/v1/Products ('{id}') — получить номенклатуру по идентификатору.

PUT /api/v1/Products ('{id}') — добавить/ редактировать номенклатуру по идентификатору.

Обновление справочника со сбросом всех записей

POST /api/v1/Products/BeginOverwrite — начинает процедуру пакетной выгрузки номенклатуры на сервер. Все позиции будут накапливаться и не будут доступны до вызова функции EndOverwrite.

POST /api/v1/Products/EndOverwrite — завершает процедуру пакетной выгрузки номенклатуры. Только после вызова этой функции сервер завершит обработку переданных позиций номенклатуры и они попадут в справочник товаров. Старый справочник товаров при этом будет полностью очищен.

Обновление записей

POST /api/v1/Products/BeginUpdate — начинает процедуру пакетного обновления номенклатуры на сервере. Все позиции будут накапливаться и не будут доступны до вызова функции EndUpdate.

POST /api/v1/Products/EndUpdate — завершает процедуру пакетного обновления номенклатуры. Только после вызова этой функции сервер завершит обработку переданных позиций номенклатуры и они попадут в справочник товаров. Выгруженная номенклатура будет слита с существующим на сервере справочником.

Сброс обновления/ перезаписывания

POST /api/v1/Products/ResetUpdate — сбрасывает процедуру пакетного обновления номенклатуры.

Обновление номенклатуры таблицей значений

POST /api/v1/Products/BeginUploadProducts — начинает выгрузку позиций номенклатуры.

POST /api/v1/Products/AddProductToUpload — добавляет в выгрузку товаров товар с упаковкой.

POST /api/v1/Products/AddProductsToUpload — добавляет в выгрузку товаров товарs с упаковками.

POST /api/v1/Products/EndUploadProducts — завершает выгрузку товаров.

Таблицы

Работа с массивом

GET /api/v1/Tables/BiznesProcessy — получить все записи таблицы.

Работа с записями по идентификатору

POST /api/v1/Tables/BiznesProcessy — редактировать/ добавить запись.

DELETE /api/v1/Tables/BiznesProcessy ('{uid}') — удалить запись из таблицы.

GET /api/v1/Tables/BiznesProcessy ('{uid}') — получить запись по идентификатору.

PATCH /api/v1/Tables/BiznesProcessy ('{uid}') — редактировать запись.

PUT /api/v1/Tables/BiznesProcessy ('{uid}') — редактировать/добавить запись по известному идентификатору.

Обновление справочника со сбросом всех записей

POST /api/v1/Tables/BiznesProcessy/BeginOverwrite — начинает процедуру пакетной выгрузки строк таблицы на сервер. Все позиции будут накапливаться и не будут доступны до вызова функции EndOverwrite.

POST /api/v1/Tables/BiznesProcessy/EndOverwrite — завершает процедуру пакетной выгрузки строк таблицы. Только после вызова этой функции сервер завершит обработку переданных позиций и они попадут в таблицу. Старое содержимое при этом будет полностью очищено.

Обновление записей

POST /api/v1/Tables/BiznesProcessy/BeginUpdate — начинает процедуру пакетного обновления строк таблицы на сервере. Все передаваемые будут накапливаться и не будут доступны до вызова функции EndUpdate.

POST /api/v1/Tables/BiznesProcessy/EndUpdate — завершает процедуру пакетного обновления строк таблицы. Только после вызова этой функции сервер завершит обработку переданных позиций и они попадут таблицу. Выгруженные позиции будут слиты с существующей на сервере таблицей.

Сброс обновления\перезаписывания

POST /api/v1/Tables/BiznesProcessy/ResetUpdate — сбрасывает процедуру пакетного обновления строк таблицы.

Документы

Получение списка типов документов

GET /api/v1/DocTypes — список типов документов.

GET /api/v1/DocTypes ('{uni}') — получить тип документа по идентификатору.

Получение всех полей типа документа

https://localhost:9000/api/v1/DocTypes?$expand=allfields

Работа с массивом

GET /api/v1/Docs — список документов.

POST /api/v1/Docs редактировать/ добавить документ.

Работа с записями по идентификатору

DELETE /api/v1/Docs ('{id}') — удалить документ.

GET /api/v1/Docs ('{id}') — получить документ по идентификатору.

PATCH /api/v1/Docs ('{id}') — редактировать документ.

PUT /api/v1/Docs ('{id}') — редактировать/добавить документ по известному идентификатору.

Обновление записей

При любом редактировании документа, он сразу не сохраняется в систему. Сохранение происходит, если вызвать принудительное сохранение (EndUpdate), либо через 30 сек от последнего изменения.

POST /api/v1/Docs ('{id}')/EndUpdate — принудительно сохраняет документ, когда все строки уже загружены (не дожидаясь сохранения через 30 сек, как указано выше).

Получение строк документа

GET /api/v1/Docs ('{key}')/declaredItems — получить строк документа.

POST /api/v1/Docs ({key})/declaredItems — редактировать/добавить строку документа.

https://localhost:5001/api/v1/Docs ('{id}') ?$expand=currentItems — получить CurentItems строк документа по его ID.

Блокировка документа

POST /api/v1/Docs ('{id}')/Block — блокирует документ для совместной работы.

Тело запроса:

{"timeout»:1000}

, где timeout — время блокировки документа

POST /api/v1/Docs ('{id}')/Unblock — разблокирует документ для совместной работы.

Склады

Работа с массивом

GET /api/v1/Warehouses — получить список складов.

Работа с записями по идентификатору

POST /api/v1/Warehouses — добавление/ редактирование склада.

DELETE /api/v1/Warehouses ('{id}') — удаление склада.

GET /api/v1/Warehouses ('{id}') — получить конкретный склад.

PATCH /api/v1/Warehouses ('{id}') — редактирование существующего склада.

PUT /api/v1/Warehouses ('{id}') — добавление/ редактирование склада по существующему идентификатору.

Warehouse {

storageId (string, *optional*): Идентификатор склада, для хранения (не меняется) ,

id (string, *optional*): Уникальный идентификатор склада ,

name (string, *optional*): Наименование склада ,

cells (Array[Cell], *optional*, read only): Коллекция ячеек склада

}

Cell {

barcode (string, *optional*): Штрихкод ячейки. Может быть шаблонированным. Подробнее про применение шаблонов для ячеек, смотрите Руководство разработчика ,

id (string, *optional*): Id ячейки, искусственный ключ ,

name (string, *optional*): Наименование ячейки ,

description (string, *optional*): Описание ячейки

}

Получение информации о подключенных устройствах

GET /api/v1/Devices — список устройств.

DeviceInfo {

appInstanceId (string, *optional*),

deviceId (string, *optional*): Уникальный идентификатор терминала. Заполняется самим устройством при регистрации его в системе ,

batteryStatus (string, *optional*): Уровень заряда батарей ,

lastInfoTime (string, *optional*): Последнее время получения информации о терминале ,

userId (string, *optional*): Id пользователя, работавший с устройством в момент последнего получения информации о нем ,

warehouseId (string, *optional*): Идентификатор склада, на котором работает пользователь ,

documentId (string, *optional*): Идентификатор документа, с которым работает пользователь ,

cellId (string, *optional*): Идентификатор последней ячейки, с которой работал пользователь ,

deviceName (string, *optional*): Пользовательское имя устройства ,

documentTypeName (string, *optional*),

serverHostedDocument (boolean, *optional*),

deviceIP (string, *optional*),

userGroupId (string, *optional*)

}

Настройки

GET /api/v1/CustomSettings — список настроек

Результат:

```
{
"@odata.context": "http://localhost:9000/MobileSMARTS/api/v1/$metadata#CustomSettings&quot;,
"value": [
{
"name": "testProp",
"value": "testValue"
}
]
 }
```


POST /api/v1/CustomSettings — Добавить/отредактировать настройку

Тело запроса:


```
{
"name": "testProp",
"value": "testValue"
}
```


Также можно добавлять массив настроек:


```
 {
"value":[
 {
"name": "testProp",
"value": "testValue"
 },
 {
"name": "testProp1",
"value": "testValue1"
 }
]
 }
```


GET /api/v1/CustomSettings ('{name}') — получить настройку

Тело ответа:


```
{
"@odata.context":
"http://localhost:9000/MobileSMARTS/api/v1/$metadata#CustomSettings/$entity&quot;,
"name": "testProp",
"value": "testValue"
 }
```


PUT /api/v1/CustomSettings ('{name}') — Добавить/отредактировать настройку по известному идентификатор.

Тело запроса:

```
{
"name": "testProp",
"value": "testValue"
 }
```

DELETE /api/v1/CustomSettings ('{name}') — удалить настройку

Сообщения

Действия с сообщениями:

GET /api/v1/Messages — список сообщений.

POST /api/v1/Messages — добавить/отредактировать сообщение.

GET /api/v1/Messages ('{id}') — получить сообщение по идентификатору.

MessageInfo {

appInstanceId (string, *optional*),
deviceId (string, *optional*): Уникальный идентификатор терминала. Заполняется самим устройством при регистрации его в системе ,
batteryStatus (string, *optional*): Уровень заряда батарей ,
lastInfoTime (string, *optional*): Последнее время получения информации о терминале ,
userId (string, *optional*): Id пользователя, работавший с устройством в момент последнего получения информации о нем ,
warehouseId (string, *optional*): Идентификатор склада, на котором работает пользователь ,
documentId (string, *optional*): Идентификатор документа, с которым работает пользователь ,
cellId (string, *optional*): Идентификатор последней ячейки, с которой работал пользователь ,
deviceName (string, *optional*): Пользовательское имя устройства ,
documentTypeName (string, *optional*),
serverHostedDocument (boolean, *optional*),
deviceIP (string, *optional*),
userGroupId (string, *optional*)

}

REST API

Не нашли что искали?

Задать вопрос в техническую поддержку

Особенности загрузки данных через REST API

Последние изменения: 2024-03-26

Отложенное внесение изменений

Обновление данных с помощью REST API всегда имеет отложенное срабатывание. Это значит, что большинство справочников (номенклатура, таблицы) и документы в базе данных на сервере обновляются не мгновенно, а с задержкой (30 секунд).

Для чего это делается:

- Для предотвращения мгновенной отправки измененного справочника на мобильное устройство, так как требуется время, чтобы успеть обновить его версию на мобильном устройстве.
- Для того, чтобы избежать генерации лишнего промежуточного трафика из-за порционности POST-запросов.
- Относительно справочника номенклатуры это решает проблему лишней нагрузки на сервер по формированию справочника и его индексов, так как данный справочник имеет специфический внутренний формат (используемый для ТСД на Windows CE).

Обновление документа

При загрузке большого числа строк в документе включается режим обновления (BeginUpdate), который завершится автоматически через 30 секунд и документ сохранится на диск. В режиме обновления документ не сохраняется при добавлении каждой строки. Если после загрузки последней строки требуется сразу записать документ на диск, необходимо выполнить запрос EndUpdate. Такая схема предотвращает выполнения события на сервере до завершения загрузки всех строк документа.

Пример:

/api/v1/Docs/ChekKorrekcii ('{id}')/EndUpdate

Обновление таблицы и справочника номенклатуры

При загрузке большого числа строк в какую-либо таблицу системы необходимо включить режим обновления таблицы. Существуют 2 режима обновления — добавление новых записей и полная перезапись таблицы. При перезаписи старые данные заменяются новыми, т. е. таблица очистится и в нее добавятся новые записи.

Для перезаписи необходимо вызвать запросы:

/api/v1/Tables/Kontragenty/BeginOverwrite — включить режим перезаписи для таблицы Kontragenty

{Далее необходимо отправить все новые записи}

/api/v1/Tables/Kontragenty/EndOverwrite — завершить режим перезаписи и заменить предыдущую таблицу Kontragenty

Для обновления записей таблицы необходимо вызвать запросы:

/api/v1/Tables/Kontragenty/BeginUpdate — начать обновление таблицы

{Далее необходимо отправить все новые записи}

/api/v1/Tables/Kontragenty/EndUpdate

Для справочника номенклатуры запросы формируются следующим образом:

/api/v1/Products/BeginOverwrite

/api/v1/Products/EndOverwrite

/api/v1/Products/BeginUpdate

/api/v1/Products/EndUpdate

Пакетная загрузка номенклатуры на сервер REST API

Пакетную загрузку номенклатуры можно реализовать следующим способом:

POST /api/v1/Products/ResetUpdate — сбрасываем процедуру пакетного обновления номенклатуры на сервере

Тело сообщения:

```
{}
```

POST /api/v1/Products/BeginUpdate — начинаем процедуру пакетного обновления номенклатуры на сервере

Тело сообщения:

```
{}
```

POST /api/v1/Products — Добавляем номенклатуру на сервер двумя пакетами

Пакет 1. Тело сообщения:

```
{
 "value": [
  {
   "id": "1f738be6-0433-4a85-866f-3479f7ec1eda",
   "name": "Товар1",
   "barcode": "1001",
   "basePackingId": "шт",
   "packings": [
    {
     "id": "шт"
    }
   ]
  },
  {
   "id": "2f738be6-0433-4a85-866f-3479f7ec1eda ",
   "name": "Товар2",
   "barcode": "1002",
   "basePackingId": "шт",
   "packings": [
    {
     "id": "шт"
    }
   ]
  }
 ]
}
```


Пакет 2. Тело сообщения:

```
{
 "value": [
  {
   "id": "3f738be6-0433-4a85-866f-3479f7ec1eda ",
   "name": "Товар3",
   "barcode": "1003",
   "basePackingId": "шт",
   "packings": [
    {
     "id": "шт"
    }
   ]
  },
  {
   "id": "4f738be6-0433-4a85-866f-3479f7ec1eda ",
   "name": "Товар4",
   "barcode": "1004",
   "basePackingId": "шт",
   "packings": [
    {
     "id": "шт"
    }
   ]
  }
 ]
}
```


Для тестовых сообщений сформированы сообщения в формате json по базовым полям в количестве 2 единиц в одном пакете

POST /api/v1/Products/EndUpdate — завершаем процедуру пакетного обновления номенклатуры на сервере

Тело сообщения:

```
{}
```

интеграция, REST API

Не нашли что искали?

Задать вопрос в техническую поддержку

Расширение API через коннектор

Последние изменения: 2024-03-26

Есть возможность расширить API, создав свой коннектор в серверной части платформы.

Более подробно о создании коннектора можно посмотреть здесь.

Для этого при создании коннектор нужно унаследовать и от интерфейса IApiExtenderPlugin, например:

```
public class ApiExtenderConnector : IConnector, IApiExtenderPlugin
{
public IList<IApiGroup> ApiGroup { get; private set;}
}
```

Свойство ApiGroup возвращает массив классов, унаследованных от IApiGroup, которые будут обрабатывать все запросы расширения из внешней системы.

```
public interface IApiGroup
{
string Id { get; }
object CreateNewEntity();
string GetEntityIdFieldName();
IEnumerable<object> GetList();
object Get(string id);
object Post(object input);
object Put(string id, object input);
void Delete(string id);
}
```

Id — идентификатор, который должен быть уникальным в системе. Он может содержать как буквы, так и цифры. Для обращения к этому расширению в url строку нужно будет вставить этот идентификатор, например:

Создаем класс коннектора:

```
 public class ApiExtenderConnector : IConnector, IApiExtenderPlugin
{
public ApiExtenderConnector() {
ApiGroup = new List<IApiGroup>();
ApiGroup.Add(new ApiTestGroup());
}
public bool IsSelfTimeoutBehavior { get { return false; } }
private string id;
[System.ComponentModel.Description("Идентификатор.")]
public string Id
{
```

```
get { return this.id; }
set { this.id = value; }
}
bool initialized = false;
public bool Initialized
{
get
{
return this.initialized;
}
}
private int timeout = 0;
public int Timeout
{
get { return this.timeout; }
set { this.timeout = value; }
}
private bool enabled = true;
public bool Enabled
{
get
{
return this.enabled;
}
set
{
this.enabled = value;
}
}
private TimeoutBehavaior timeoutBehavaior = TimeoutBehavaior.ThrowException;
public TimeoutBehavaior TimeoutBehavaior
{
get { return this.timeoutBehavaior; }
set { this.timeoutBehavaior = value; }
}
public bool IsSupportDeviceInfoInArgs
{get { return false; }
}
[Cleverence.DataCollection.Xml.XmlSerializable(Cleverence.DataCollection.Xml.XmlSerializationType.
None)]
public IList<IApiGroup> ApiGroup { get; private set;}
public void CheckLicenseLimitations(List<LicenseExternalSystem> supportedSystems)
{
}
public void Initialize()
{
}
public object InvokeMethod(string methodName, object[] args)
{
return null;
}
}
```

Далее необходимо откомпилировать и положить готовую библиотеку в папку сервера. В результате после удачного запуска сервера проверяем ответ сервера по пути

https://localhost:<порт базы>/api/v1/

В ответе должна содержаться группа

{"name»:"Plugins/ApiTestGroup»,"kind»:"EntitySet»,"url»:"Plugins/ApiTestGroup"}

Далее при переходе по пути

https://localhost:<порт базы>/api/v1/Plugins/ApiTestGroup

получаем список наших сущностей (результат выполнения функции GetList ())


```
 {
 "@odata.context":"https://localhost:9005/MobileSMARTS/api/v1/$metadata#Plugins&quot;,
 "value":[
 {"id":"1","name":"1111","valueInt":0,"valueBool":false},
 {"id":"2","name":"2223","valueInt":0,"valueBool":false},
 {"id":"3","name":"3333","valueInt":0,"valueBool":false}]
}
```

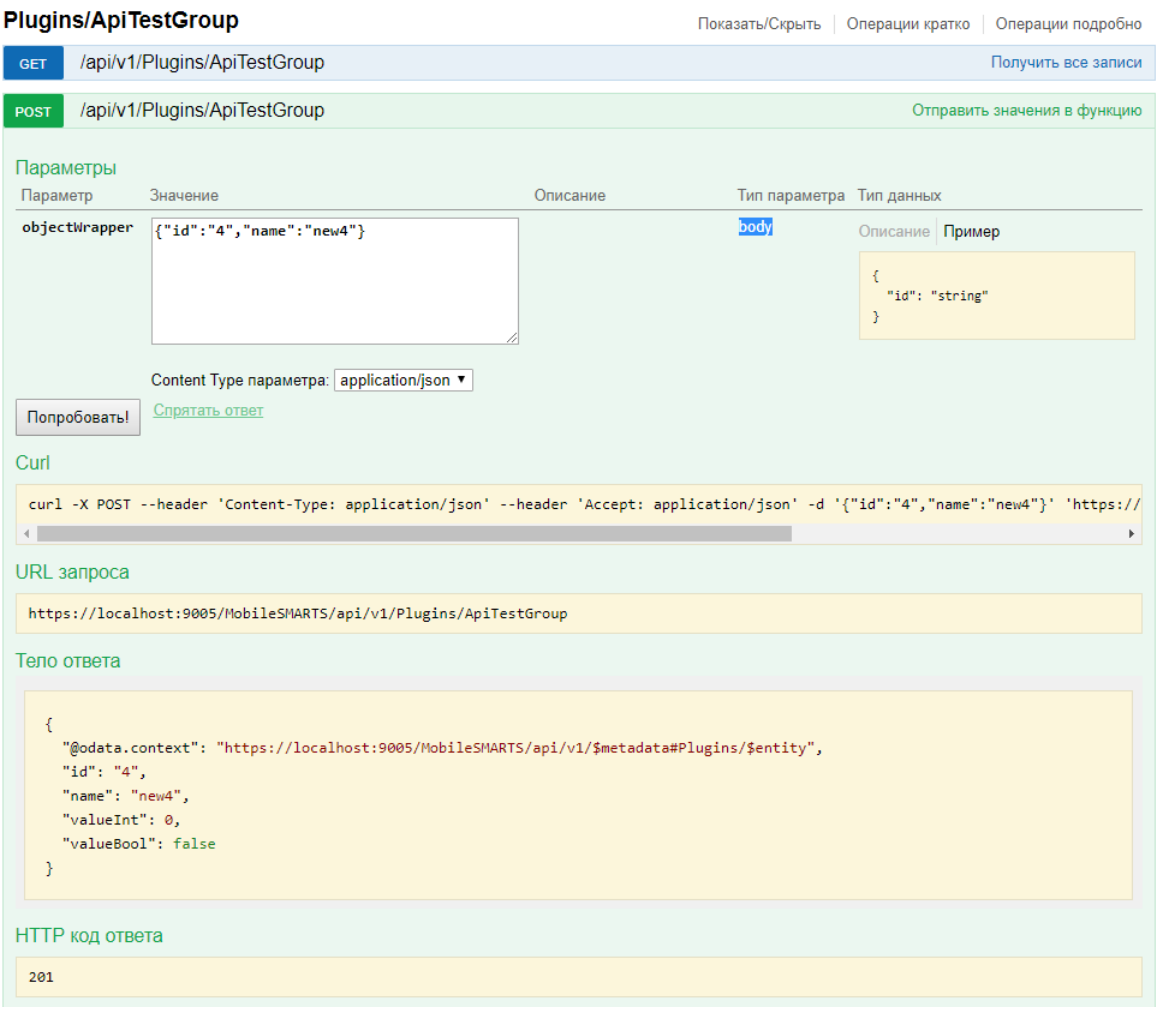

Далее POST запрос можем проверить в swagger. Для этого заходим

https://localhost:<порт базы>/swagger

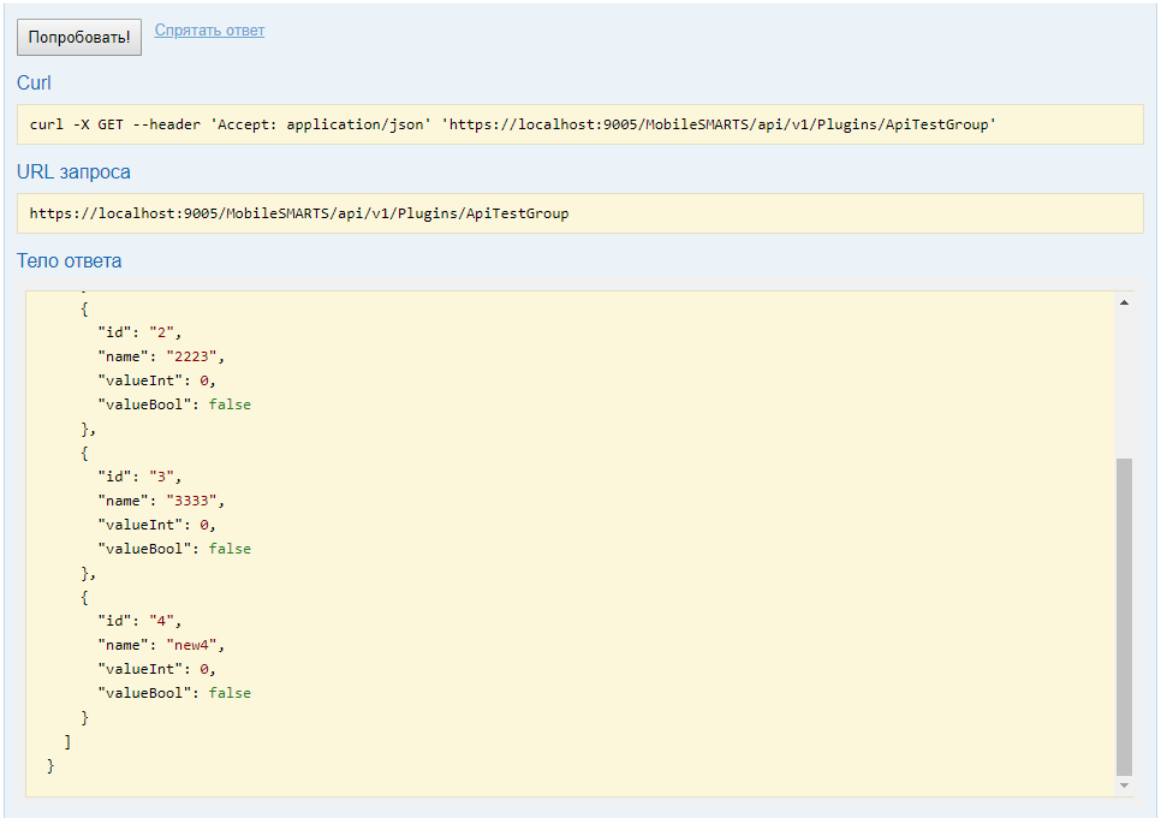
Находим там нашу группу Plugins/ApiTestGroup, выбираем POST, вставляем в параметр body:

{"id»:"4»,"name»:"new4"}

Нажимаем кнопку «Попробовать!»



Далее заходим в блок GET и видим что наша новая сущность сохранилась:



Для удаления заходим в блок DELETE, вводим наш новый идентификатор 4 и выполняем.

DELETE /api/v1/Plugins/ApiTestGroup('{id}') Запрос на удаление значений

Параметры

Параметр	Значение	Описание	Тип параметра	Тип данных
id	4	key: id	path	string
If-Match	*	If-Match header	header	string

Что может прийти в ответ

HTTP код	Причина	Структура ответа	Заголовки
204	NoContent		

Попробовать!

Проверяем в блоке GET и видим, что новая сущность исчезла.

Попробовать! Спрятать ответ

Curl

```
curl -X GET --header 'Accept: application/json' 'https://localhost:9005/MobileSMARTS/api/v1/Plugins/ApiTestGroup'
```

URL запроса

```
https://localhost:9005/MobileSMARTS/api/v1/Plugins/ApiTestGroup
```

Тело ответа


```
    {
      "id": "2",
      "name": "2223",
      "valueInt": 0,
      "valueBool": false
    },
    {
      "id": "3",
      "name": "3333",
      "valueInt": 0,
      "valueBool": false
    }
  ]
}
```


Не нашли что искали?

Задать вопрос в техническую поддержку

Блокировка документа через REST API

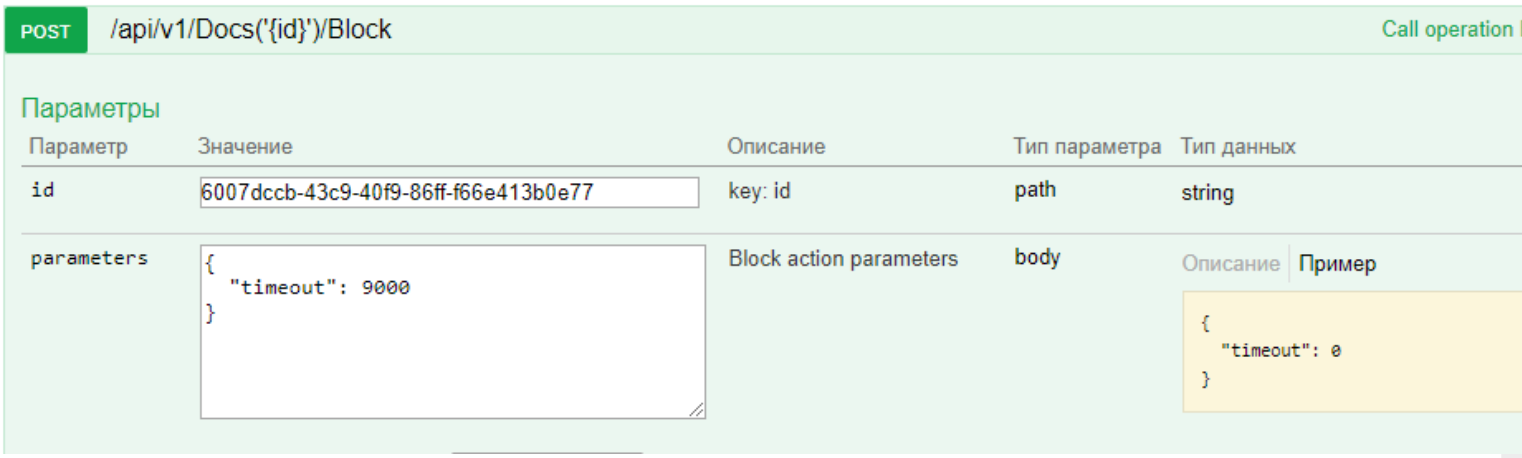
Последние изменения: 2024-03-26

Для редактирования коллективного документа через API нужно его заблокировать.

Блокировка методом block

Для этого нужно использовать метод block с идентификатором документа

http://localhost:12471/MobileSMARTS/api/v1/Docs ('6007dccb-43c9-40f9-86ff-f66e413b0e77')/Block

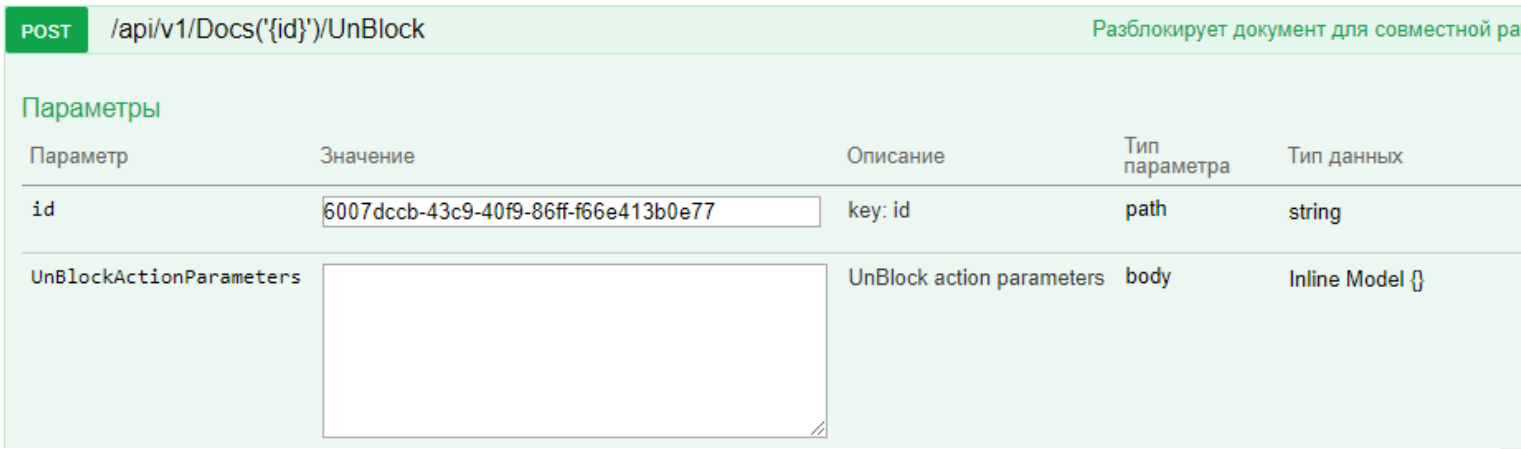


В параметрах нужно обязательно указывать таймаут (через какое время документ автоматически разблокируется, если его не разблокировали вручную). Диапазон от 500 до 10000 мс.

Разблокировка методом unblock

Для ручной разблокировки нужно использовать метод unblock с указанием идентификатора документа. После ручной разблокировки — автоматическая разблокировка не произойдет.

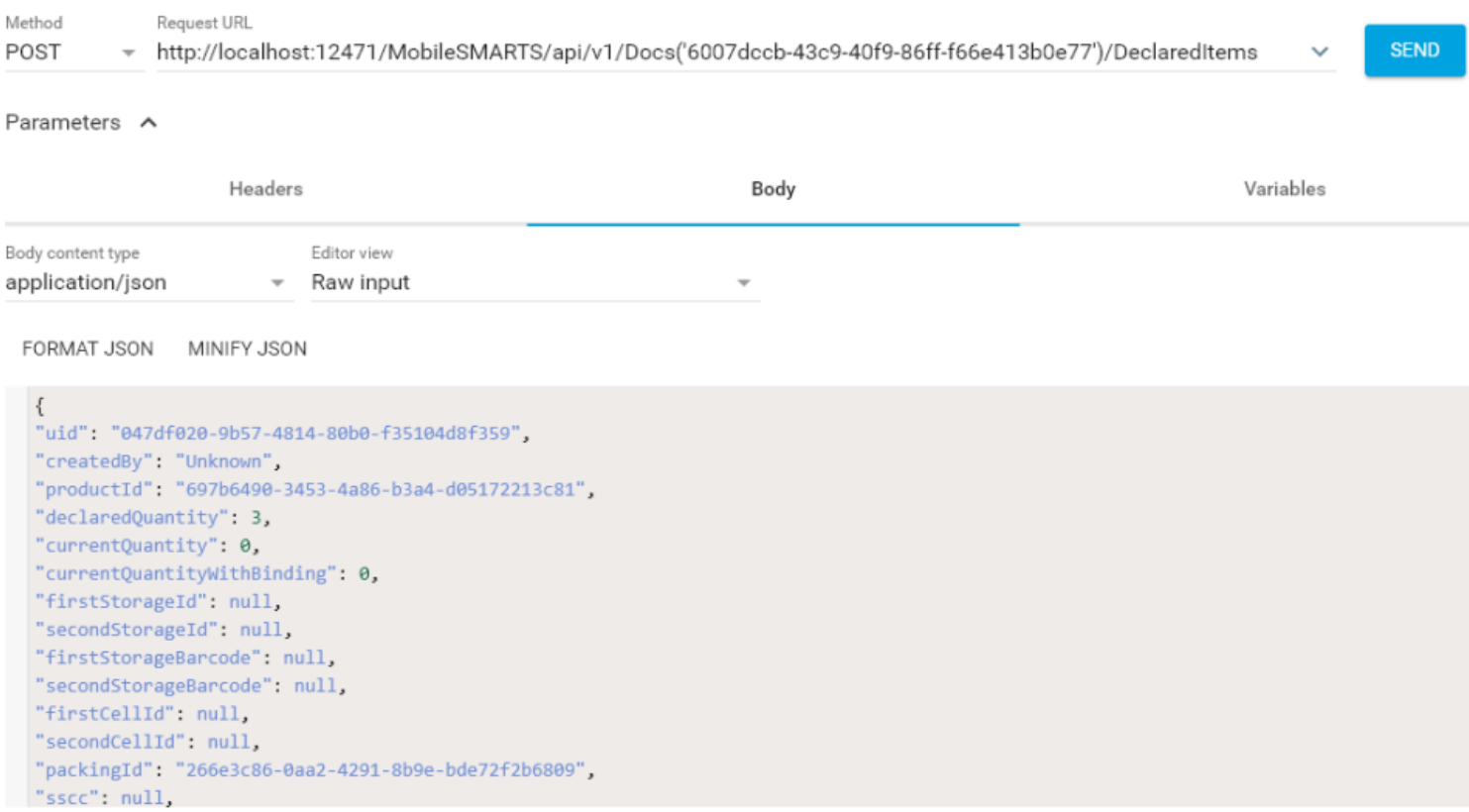
http://localhost:12471/MobileSMARTS/api/v1/Docs ('6007dccb-43c9-40f9-86ff-f66e413b0e77')/Unblock



Добавления новой строки методом post

Для добавления новой отдельной строки документа можно использовать метод post в DeclaredItems.

Пример

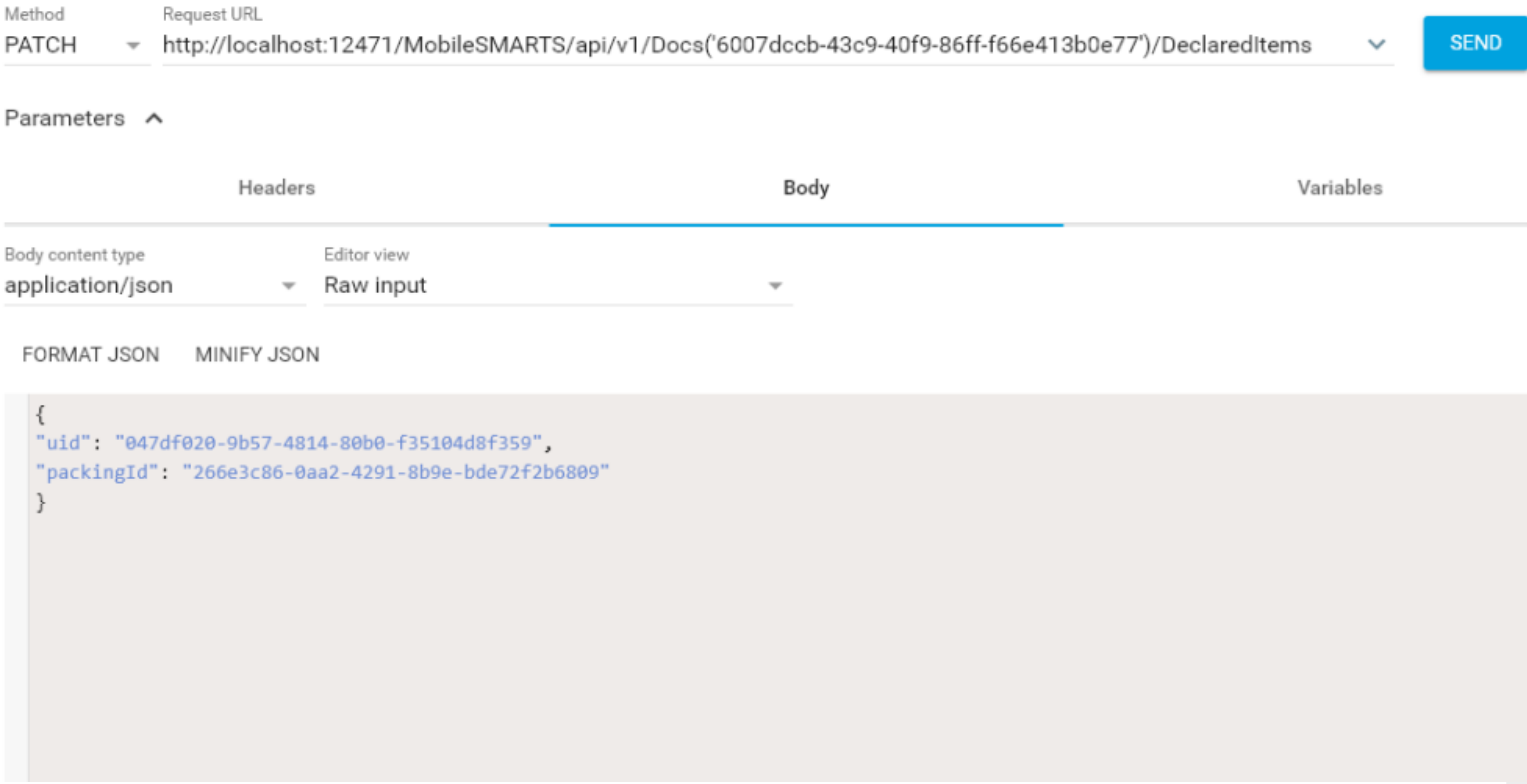


Редактирование методом patch

Для редактирования существующей строки можно пользоваться методом patch.

Пример

http://localhost:12471/MobileSMARTS/api/v1/Docs ('6007dccb-43c9-40f9-86ff-f66e413b0e77')/DeclaredItems



При патче можно указывать только те поля, которые необходимо обновить в нужной строке. В теле обязательно нужно указывать uid нужной строки!

Удаление строки методом Delete

Для удаления нужной строки у документа можно использовать метод Delete.

Пример

DELETE http://localhost:12471/MobileSMARTS/api/v1/Docs ('6007dccb-43c9-40f9-86ff-f66e413b0e77')/DeclaredItems ('047df020-9b57-4814-80b0-f35104d8f359'),

тут в урле необходимо указать как идентификатор документа, так и uid строки из declaredItems.

Не нашли что искали?

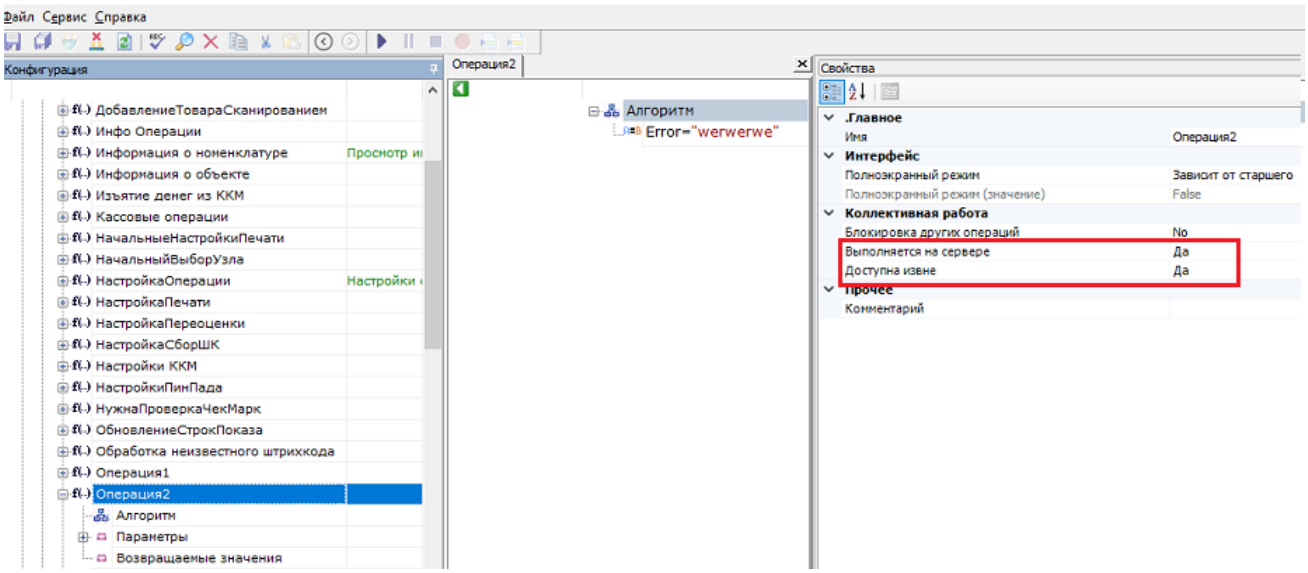
Задать вопрос в техническую поддержку

Вызов серверных операций через REST API

Последние изменения: 2024-03-26

Начиная с версии 3.2.46.20453 у платформы Mobile SMARTS появилась возможность вызывать серверные операции с помощью REST API. Для этого необходимо:

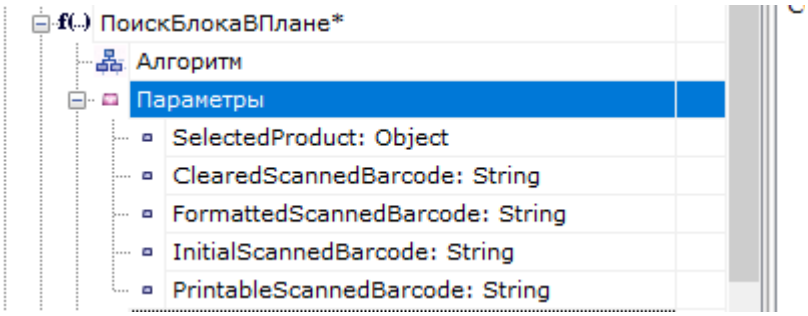
1. В панели управления Mobile SMARTS для требуемой операции установить значения «Да» у параметров «Выполняется на сервере» и «Доступна извне».



2. Выполнить запрос http://localhost:9000/MobileSMARTS/api/v1/Operations('{OperationName}'),

   где {OperationName} - имя требуемой операции.

3. Входные параметры, необходимые для выполнения операции указываются тут:



4. Если в операцию необходимо передать параметры, то они перечисляются в теле запроса в формате json:

```
{
"a": "параметр1",
"b": "параметр2"
}
```

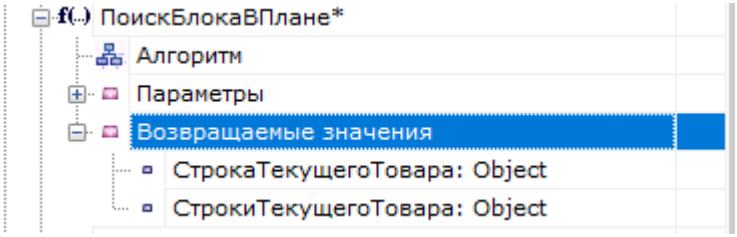
5. Если в параметрах необходимо передать сложный тип, то при описании параметра в json необходимо добавить поле "@odata.type", в котором указывается тип передаваемого параметра. Например:

```
{
"field0": "11",
"field1": {
"@odata.type": "#Cleverence.Warehouse.Document",
"id": "7c342252-de63-42fe-9742-b47b0a40a7ee"
}
}
```

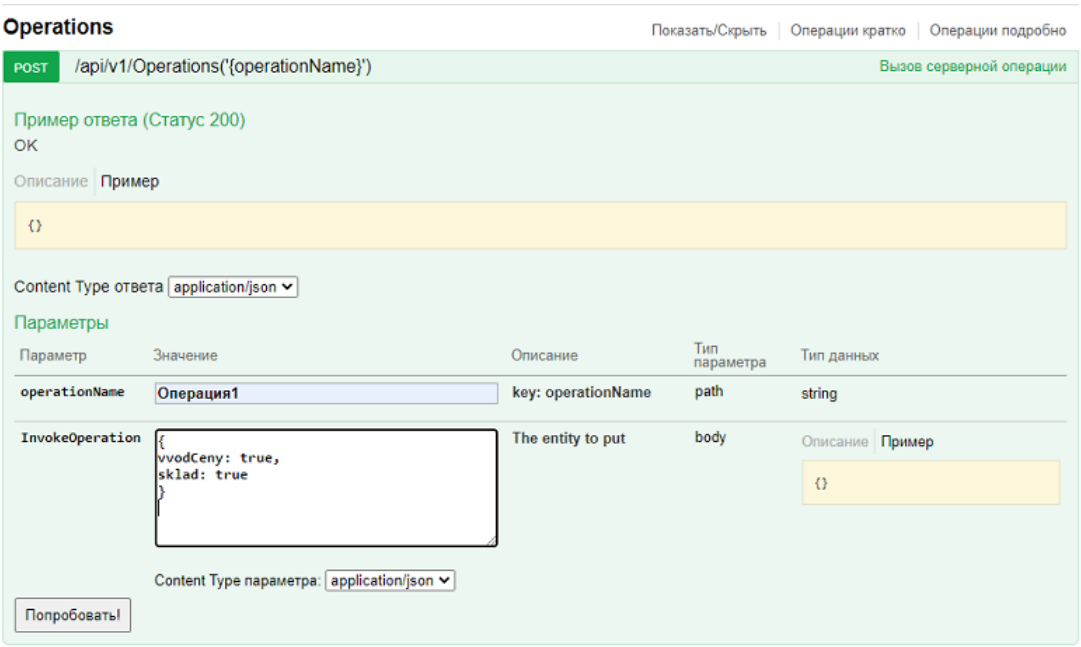
6. **В ответе сервер вернет результат выполнения операции в виде json, в котором будут перечислены выходные параметры после выполнения операции. Например:**

```
{
vvodCeny: true,
sklad: true
}
```

7. **Выходные параметры указываются при настройке операции тут:**



8. **Выполнение серверной операции можно протестировать в Swagger:**



Не нашли что искали?

Задать вопрос в техническую поддержку