RuBackup
Система резервного копирования и восстановления данных

Руководство по установке и взаимодействию с программным интерфейсом RuBackup REST API

Версия 2.3.0

25.10.2024

Содержание

Введение

Система резервного копирования и восстановления данных RuBackup предоставляет пользователю возможность взаимодействия с активным сервером резервного копирования посредством HTTP-запросов к его ресурсам.

Программный интерфейс RuBackup реализован и документирован с использованием набора инструментов Swagger. Для описания REST API Swagger использует формат JSON. Swagger используется вместе с набором программных средств с открытым исходным кодом для проектирования, создания, документирования и использования веб-служб REST.

Настоящее руководство описывает базовые шаги установки, настройки и эксплуатации RuBackup REST API. Руководство предназначено для системных администраторов, отвечающих за сопровождение СРК.

Перед установкой RuBackup REST API

Базовые требования

Перед инсталляцией RuBackup API необходимо убедиться, что выполнены все действия для установки СРК RuBackup согласно документации «Руководстве по установке серверов резервного копирования и Linux клиентов»:

1. Скачаны все необходимые пакеты актуальной версии СРК: rubackup-common, rubackup-client, rubackup-server и rubackup-rest-api;

2. Пакеты rubackup-common, rubackup-client, rubackup-server установлены;

3. Проведена настройка основного сервера с помощью интерактивной утилиты rb_init.

Подробнее о вариантах установки REST API можно прочитать в разделе
.

Также проверьте наличие файлов-сертификатов RuBackup, которые используются программным интерфейсом для создания защищённого соединения.

Расположение сертификатов в файловой системе:

1. /opt/rubackup/keys/server/serverCert.crt

2. /opt/rubackup/keys/server/serverKey.key

3. /opt/rubackup/keys/client/clientCert.crt

4. /opt/rubackup/keys/client/clientKey.key

5. /opt/rubackup/keys/rootCA/serverRootCACert.crt

Особенности установки пакетов в Linux

Дистрибутив RuBackup REST API поставляется в виде deb и rpm-пакетов. Для разных дистрибутивов Linux, по причине их отличий друг от друга, предусмотрены специально подготовленные пакеты RuBackup.

В зависимости от типа используемого пакетного менеджера в вашем дистрибутиве Linux, процедура установки и удаления пакетов может использовать команды `dpkg`, `rpm`, `apt`, `yum` и пр. В настоящем руководстве процедуры установки описаны для пакетного менеджера, который оперирует пакетами deb. Например, команда установки пакета в операционной системе Ubuntu 20.04 выглядит следующим образом:

```
$ sudo dpkg -i rubackup-rest-api.deb
```

Для установки API в ОС с пакетным менеджером, который оперирует rpm-пакетами, вместо вышеуказанной команды следует выполнить команду:

```
$ sudo rpm -i rubackup-rest-api.rpm
```

Процедуры удаления пакетов в настоящем руководстве описаны для пакетного менеджера, который оперирует пакетами deb. Например, процедура удаления пакета RuBackup API выглядит следующим образом:

```
$ sudo apt remove rubackup-rest-api
```

Для удаления RuBackup API в операционной системе с пакетным менеджером, который оперирует rpm пакетами, вместо вышеуказанной команды следует выполнить:

```
$ sudo yum remove rubackup-rest-api
```

либо:

```
$ sudo rpm -e rubackup-rest-api
```

Некоторые операционные системы, например Alt Linux, используют пакетную систему rpm, но вместо yum используют apt. Перед установкой или удалением пакета RuBackup REST API следует уточнить, какие команды необходимо использовать для вашего дистрибутива Linux.

Установка RuBackup REST API

Перед использованием rubackup-rest-api рекомендуется внести изменения в конфигурационный файл postgresql.conf для увеличения количества зарезервированных подключений суперпользователя, например, командой:

```
nano /etc/postgresql/<номер версии>/main/postgresql.conf
```

где <номер версии> — это номер версии postgresql

Необходимо выставить значение:

```
superuser_reserved_connections = 50
```

Для инсталляции RuBackup API установите пакет rubackup-rest-api, например, командой:

```
$ sudo dpkg -i rubackup-rest-api.deb
```

Имя файла пакета может отличаться в зависимости от сборки.

После установки пакета вы можете сразу запустить процесс RuBackup API, если у вас уже определён FQDN для хоста и он явно указан в файле /**etc**/**hosts**.

Настройка RuBackup REST API

Настройка RuBackup API осуществляется пользователем при помощи изменения переменных окружения.

Внимание! Для RuBackup API данные из конфигурационного файла RuBackup (/opt/rubackup/etc/config.file) не используются.

Ниже представлен перечень переменных окружения доступных для изменения из файла /opt/rubackup/etc/rubackup_api.env:

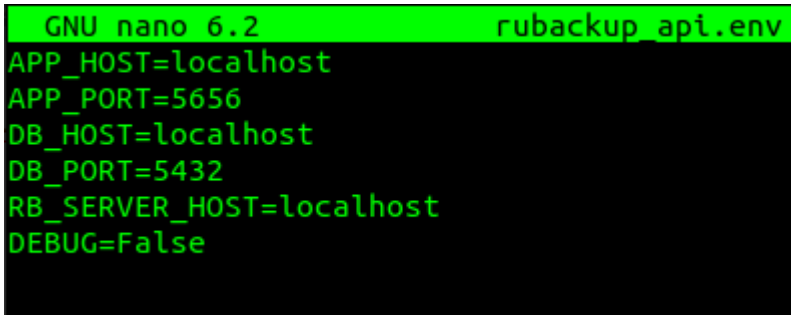
Имя переменной	Описание	Возможные значения
APP_HOST	Желаемый IP-адрес или FQDN, который будет использоваться как часть адреса сервера API. Если IP-адрес или FQDN указан некорректно, то при запуске RuBackup API не будут записываться access_token и refresh_token в cookies.	IP/FQDN (localhost)

Имя переменной	Описание	Возможные значения
APP_PORT	Желаемый порт, который будет использоваться как часть адреса сервера API	Порт (5656)
DB_HOST	IP или FQDN сервера PostgreSQL с базой данных RuBackup	IP/FQDN (localhost)
DB_PORT	Порт сервера PostgreSQL с базой данных RuBackup	Порт (5432)
RB_SERVER_HOST	IP или FQDN основного сервера RuBackup	IP/FQDN (localhost)
DEBUG	Режим расширенного логирования	True/False

Описанные переменные могут быть применены локально, через команду export. Например:

 # export APP_HOST=api.rubackup.local

Также у пользователя есть возможность зафиксировать значения переменных, глобально описав их в файле /opt/rubackup/etc/rubackup_api.env. Пример:



Запуск Swagger и Tucana будет произведён по адресу, указанному в параметре APP_HOST.

1) Для того, что запуск был произведен по доменному имени, указать нужно именно его, например:

APP_HOST=api.rubackup.local

Также для же для этого в /etc/hosts должен быть указан этот FQDN.

2) Для запуска через localhost можно оставить параметры по умолчанию:

APP_HOST=localhost

Настройка журналирования RuBackup REST API

Для обеспечения гибкости процесса журналирования действий сервера предусмотрен специальный конфигурационный файл, расположенный по пути /opt/rubackup/etc/rubackup_api_logger.conf. Лог-файлы располагаются в директории по пути /opt/rubackup/logs/rubackup-api.

Логгеры

В конфигурационном файле присутствует четыре типа логгеров:

- *logger_root* — логгер, от которого должны наследоваться все остальные. По умолчанию записывает сообщения уровня Info и выше;
- *logger_rb_api* — логирует общие ошибки и информационные сообщения по API. По умолчанию записывает сообщения уровня Info и выше. Ведёт запись в консоль и в файл;
- *logger_rb_access* — логирует инциденты, связанные с авторизацией. По умолчанию записывает сообщения уровня Info и выше. Ведёт запись в консоль и в файл.
- *logger_tornado* — логирует ошибки web-сервера Tornado. По умолчанию записывает сообщения уровня Info и выше. Ведёт запись в консоль и в файл.

Хэндлеры

Для каждого логгера в конфигурационном файле присутствуют хэндлеры, в которых написано, как и какие ошибки обрабатывать и куда их записывать.

Всего есть три хэндлера:

- *handler_file* — перехватывает сообщения уровня Info, записывает их в /opt rubackup/log/rubackup_api/access/rubackup-api.log, применяет определённое форматирование. Также после 00:00 создаёт новый файл для записи логов. Количество файлов журнала не превышает 15.

- *handler_console* — перехватывает сообщения уровня Info, применяет определённое форматирование и выводит их в консоль;

- *handler_access* — перехватывает сообщения уровня Info, применяет определённое форматирование, записывает инциденты, связанные с доступом, в /opt/rubackup/log/rubackup_api/access/rubackup-api.access.log.

Также после 00:00 создаёт новый файл для записи логов. Количество файлов журнала не превышает 15.

Форматеры

Описывают формат, в котором нужно записывать и/или отображать сообщения в файле и/или консоли.

Дополнительную информацию по журналированию можно найти по ссылке: *https://docs.python.org/3/library/logging.config.html#configuration-file-format*

Настройка запуска RuBackup REST API

После установки пакета и настройки переменных окружения можно производить запуск RuBackup API.

Сервер RuBackup API представляет собой фоновое приложение (сервис, демон).

Расположение:

/opt/rubackup/bin/rubackup_api

Запуск в терминальном режиме:

```
$ rubackup_api --start
```

Остановка:

```
$ rubackup_api --stop
```

Перезагрузка:

```
$ rubackup_api --restart
```

Для штатной эксплуатации RuBackup API рекомендуется запустить его как сервис. Для этого выполните следующие действия:

1. Включите сервис RuBackup API:

```
$ sudo systemctl enable \
    /opt/rubackup/etc/systemd/system/rubackup_api.service
```

2. Перезагрузите `systemctl`:

```
$ sudo systemctl daemon-reload
```

3. Запустите сервис `rubackup_api`:

```
$ sudo systemctl start rubackup_api.service
```

Уточнить статус RuBackup API можно при помощи команды:

```
$ systemctl status rubackup_api.service
```

rubackup_api.service - RuBackup API

```
   Loaded: loaded (/etc/systemd/system/rubackup_api.service; enabled; vendor preset:
enabled)
   Active: active (running) since Tue 2024-05-07 22:06:24 MSK; 24min ago
 Main PID: 69213 (rubackup_api)
    Tasks: 2 (limit: 9430)
   Memory: 61.0M
   CGroup: /system.slice/rubackup_api.service
           ├──69213 /bin/sh /opt/rubackup/bin/rubackup_api --start
           └──69214 /opt/rubackup/lib/rubackup_rest_api_lib/rubackup_api.bin --start
мая 07 22:06:24 rb-primary systemd[1]: Started RuBackup API.
мая 07 22:06:25 rb-primary rubackup_api[69214]: RuBackup API Logger initializing
мая 07 22:06:26 rb-primary rubackup_api[69214]: 2024-05-07 22:06:26,066 -
[WARNING] - 'The rubackup database has not been initialized. Please authenticate'
мая 07 22:06:26 rb-primary rubackup_api[69214]: 2024-05-07 22:06:26,070 -
[INFO] - 'RuBackup REST API is running on [https://rubackup.api.local:5656]'
```

Внимание! Сообщение 'The rubackup database has not been initialized. Please authenticate' является предупреждением пользователя о необходимости пройти аутентификацию хотя бы один раз для продолжения работы с сервисом. Для прохождения аутентификации воспользуйтесь методом POST /login напрямую или Tucana.

Установка RuBackup REST API на выделенный хост

Перед инсталляцией RuBackup API убедитесь, что выполнены все действия для установки СРК RuBackup согласно документу «Руководство по установке серверов резервного копирования и Linux-клиентов»:

1. Скачаны все необходимые пакеты актуальной версии rubackup-common, rubackup-client, rubackup-server и rubackup-rest-api;
2. Пакеты rubackup-common, rubackup-client, rubackup-server установлены. Проводить настройку с помощью утилиты rb_init не нужно;
3. Существует хост с установленным, настроенным и запущенным основным сервером rubackup;
4. Существует хост с базой данных rubackup.

Для инсталляции RuBackup API установите пакет rubackup-rest-api, например, командой:

```
$ sudo dpkg -i rubackup-rest-api.deb
```

Внимание! Имя файла пакета может отличаться в зависимости от сборки.

Настройка RuBackup REST API на выделенном хосте

Настройка RuBackup API осуществляется пользователем при помощи изменения переменных окружения.

Внимание! Для RuBackup API из конфигурационного файла RuBackup (/opt/rubackup/etc/config.file) данные не используются.

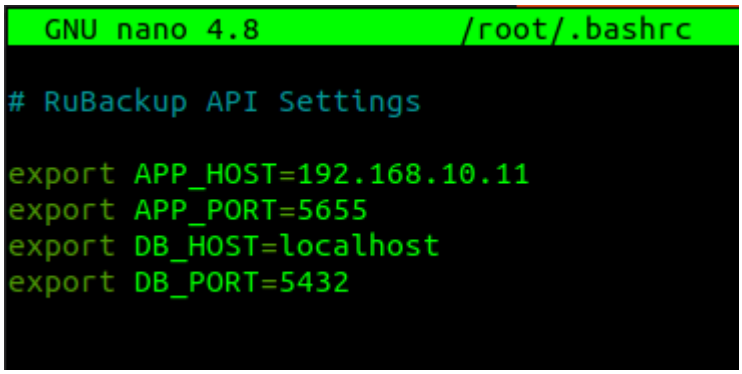
Ниже представлен перечень переменных окружения доступных для изменения из файла /opt/rubackup/etc/rubackup_api.env:

Имя переменной	Описание	Возможные значения
APP_HOST	Желаемый IP-адрес или FQDN, который будет использоваться как часть адреса сервера API. Если IP-адрес или FQDN указан некорректно, то при запуске RuBackup API не будут записываться access_token и refresh_token в cookies	IP/FQDN (localhost)
APP_PORT	Желаемый порт, который будет использоваться как часть адреса сервера API	Порт (5656)
DB_HOST	IP или FQDN сервера PostgreSQL с базой данных RuBackup	IP/FQDN (localhost)
DB_PORT	Порт сервера PostgreSQL с базой данных RuBackup	Порт (5432)
RB_SERVER_HOST	IP или FQDN основного сервера RuBackup	IP/FQDN (localhost)
DEBUG	Режим расширенного логирования	True/Fals

Описанные переменные могут быть применены локально с помощью команды export. Например:

```
# export APP_HOST=api.rubackup.local
```

Также, у пользователя есть возможность зафиксировать значения описанных переменных глобально описав их в файле ‘.bashrc’. Пример:



После этого необходимо перезагрузить переменные окружения:

```
# . .bashrc
```

Для запуска Swagger и Tucana на выделенном хосте в файл переменных окружения /opt/rubackup/etc/rubackup_api.env нужно установить следующие параметры:

APP_HOST=IP или FQDN хоста, на котором установлен и будет запущен rest-api
APP_PORT=5656
DB_HOST=IP или FQDN хоста с базой данных
DB_PORT=5432
RB_SERVER_HOST=IP или FQDN хоста основного сервера rubackup

Настройка запуска RuBackup REST API на выделенном хосте

После установки пакета и настройки переменных окружения можно производить запуск RuBackup API.

Сервер RuBackup API представляет собой фоновое приложение (сервис, демон).

Расположение:

/opt/rubackup/bin/rubackup_api

Запуск в терминальном режиме:

```
$ rubackup_api --start
```

Остановка:

```
$ rubackup_api --stop
```

Перезагрузка:

```
$ rubackup_api --restart
```

Для штатной эксплуатации RuBackup API рекомендуется запустить его как сервис. Для этого выполните следующие действия:

4. Включите сервис RuBackup API:

```
$ sudo systemctl enable \
      /opt/rubackup/etc/systemd/system/rubackup_api.service
```

5. Перезагрузите systemctl:

```
$ sudo systemctl daemon-reload
```

6. Запустите сервис `rubackup_api`:

```
$ sudo systemctl start rubackup_api.service
```

Уточнить статус RuBackup API можно при помощи команды:

```
$ systemctl status rubackup_api.service
```

```
rubackup_api.service - RuBackup API
    Loaded: loaded (/etc/systemd/system/rubackup_api.service; enabled; vendor
preset: enabled)
    Active: active (running) since Tue 2024-05-07 22:06:24 MSK; 24min ago
  Main PID: 69213 (rubackup_api)
     Tasks: 2 (limit: 9430)
    Memory: 61.0M
    CGroup: /system.slice/rubackup_api.service
            ├──69213 /bin/sh /opt/rubackup/bin/rubackup_api --start
            └──69214 /opt/rubackup/lib/rubackup_rest_api_lib/rubackup_api.bin --
start

мая 07 22:06:24 rb-primary systemd[1]: Started RuBackup API.
мая 07 22:06:25 rb-primary rubackup_api[69214]: RuBackup API Logger
initializing
мая 07 22:06:26 rb-primary rubackup_api[69214]: 2024-05-07 22:06:26,066 -
[WARNING] - 'The rubackup database has not been initialized. Please
authenticate'
мая 07 22:06:26 rb-primary rubackup_api[69214]: 2024-05-07 22:06:26,070 -
[INFO] - 'RuBackup REST API is running on [https://rubackup.api.local:5656]'
```

В данном случае сообщение 'The rubackup database has not been initialized. Please authenticate' означает, что сервис успешно запущен, но еще не выполнена авторизация в Swagger или Tucana.

Использование RuBackup REST API

Этот раздел описывает процесс аутентификации, авторизации и взаимодействия пользователя с программным интерфейсом RuBackup.

Аутентификация пользователя

Перед тем как пользователь сможет обратиться к ресурсам сервера RuBackup, он должен пройти аутентификацию и получить токены доступа: access_token, refresh_token и csrf_token.

Сгенерированный access_token будет действовать в течение 15 минут с момента получения, refresh_token — 24 часа с момента получения, csrf_token действует до перезагрузки сервиса rubackup_api.

По истечении срока жизни access_token его можно перевыпустить с помощью refresh_token. Если истек срок жизни refresh_token, необходимо перевыпустить новую пару токенов с помощью логина и пароля. После перезапуска rubackup_api также необходимо перевыпустить новую пару токенов с помощью логина и пароля.

Выпуск access_token, refresh_token и csrf_token через браузер

Для получения пары токенов необходимо выполнить следующие действия:

1. Перейдите по адресу *https://<app_host>:<app_port>/api/v1/* (рисунок 1):

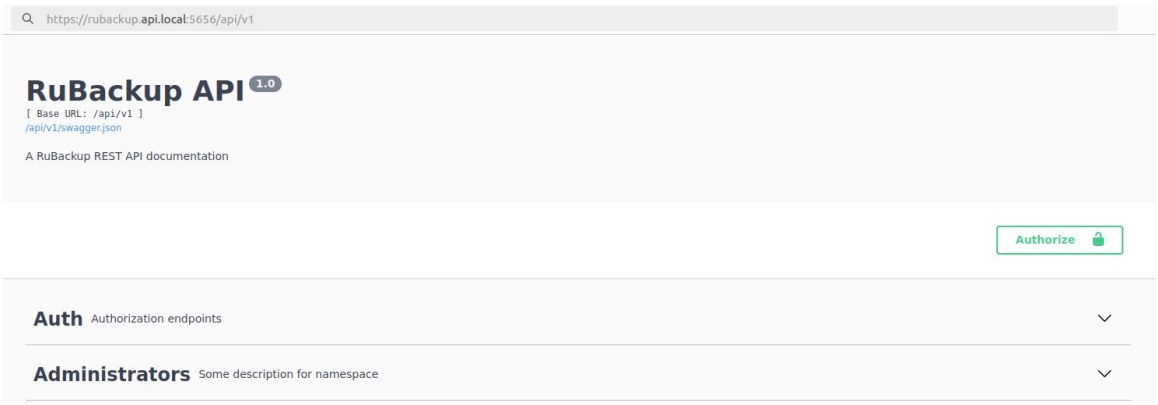


Рисунок 1

2. Перейдите на вкладку «Auth» и выберите эндпоинт «/auth/login» (рисунок 2:

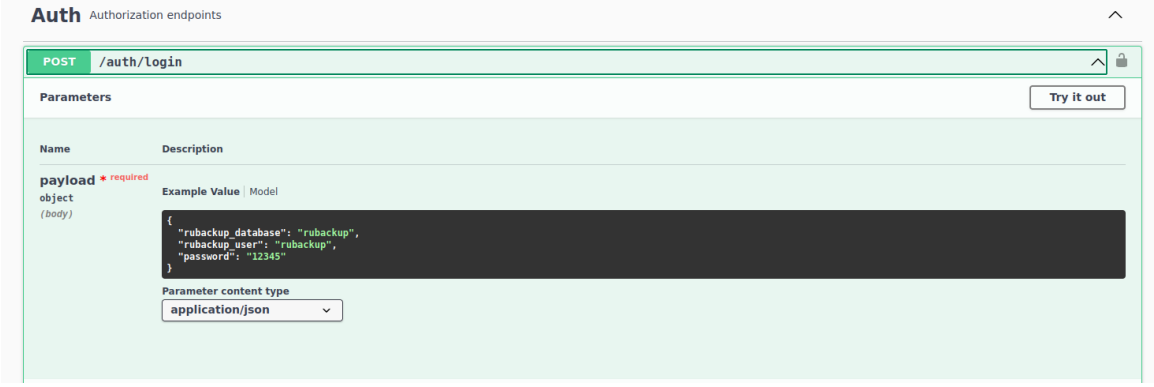


Рисунок 2

3. Нажмите кнопку «Try it out», заполните payload актуальными данными и нажмите «Execute» (рисунок 3):

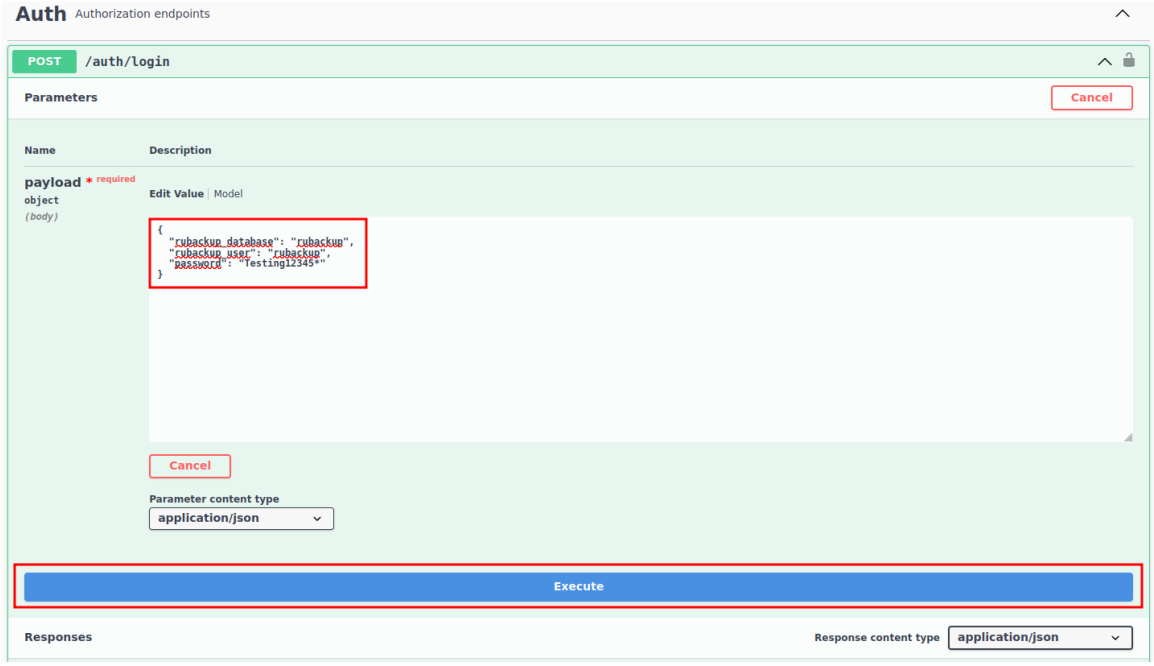


Рисунок 3

В результате проделанных операций будут получены access_token, refresh_token и csrf_token, а также сопутствующая информация о пользователе прошедшем авторизацию (рисунок 4). Также сервис автоматически разместит access_token и refresh_token в cookie-файлах.

Рисунок 4

Выпуск access_token, refresh_token и csrf_token через CLI

Чтобы произвести выпуск пары токенов через терминал, необходимо отправить POST-запрос с помощью консольной утилиты curl или любым другим удобным способом. В данном примере используется curl:

```
$ curl -k -X POST 'https://api.rubackup.local:5656/api/v1/auth/login' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{"rubackup_database": "rubackup", "rubackup_user": "rubackup",
"password": "Testing12345*"}'
```

где

https://api.rubackup.local — адрес, где запущен rubackup_api

5656 - порт, который будет использоваться как часть адреса сервера API

/api/v1/auth/login — путь до запроса на авторизацию

-H <argument> - значения, передаваемые в Headers

rubackup_database — имя служебной базы данных

rubackup_user — имя суперпользователя

password — пароль суперпользователя

Результат вернётся в формате JSON:

```
{
  "data": {
"access_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTcxNTE1ODc2MCwianRpIjoiNThjZjQ5MDAtZjE5Ni00OTZlLThkZjQtMThmZjlhY2M3Y2QxIiwidHlwZSI6ImFjY2VzcyIsInN1YiI6InJ1YmFja3VwIiwibmJmIjoxNzE1MTU4NzYwLCJjc3JmIjoiNGM4YzM0NTctNDAwNS00YzZiLTk2MWMtMGM1MDU5NjcxZTA2IiwiZXhwIjoxNzE1MTU5Nj
```

```
YwLCJmYW1pbHkiOiJ4cnByeHR3ZHJyZ3FsamN4In0.VK5K6v-0_NxSx42bU5dEMIQAYYzxn-
GTxmbjhxXjYSs",

"csrf_token": "4c8c3457-4005-4c6b-961c-0c5059671e06",

"refresh_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTcxNT
E1ODc2MCwianRpIjoiOTUzZjAxZWYtMTM4Zi00Y2ZhLTg4ODItNGI0NWQ0N2YxM2I2IiwidH
lwZSI6InJlZnJlc2giLCJzdWIiOiJydWJhY2t1cCIsIm5iZiI6MTcxNTE1ODc2MCwiY3NyZi
I6IjRjOGMzNDU3LTQwMDUtNGM2Yi05NjFjLTBjNTA1OTY3MWUwNiIsImV4cCI6MTcxNTI0NT
E2MCwiZmFtaWx5IjoieHJwcnh0d2RycmdxbGpjeCJ9.7TmdI0Cmm4knApNINDoYJuJIYdRlz
uuc1hS-1c4Y8Ws",

    "role": [

      "superuser"

    ],

    "rubackup_server_address": "10.177.xx.xxx",

    "user_name": "rubackup"

  },

  "errors": {},

  "is_error": false,

  "message": ""

}
```


Если необходимо получить еще и access_token и refresh_token из cookies, то в команде curl следует указать опцию “--cookie-jar -”, например:

```
$ curl -k --cookie-jar - -X POST
'https://api.rubackup.local:5656/api/v1/auth/login' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{"rubackup_database": "rubackup", "rubackup_user": "rubackup",
"password": "Testing12345*"}'
```

С этой опцией к выводу добавится следующая информация:


```
#HttpOnly_.rubackup.local     TRUE  /     TRUE  0   refresh_token_cookie

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTcxNTE
2MDUzNywianRpIjoiYzM4MTE0N2YtNDI0MS00Y2NjLTlhZjYtNDcwODAxMWFmOTRjIiwidHl
wZSI6InJlZnJlc2giLCJzdWIiOiJydWJhY2t1cCIsIm5iZiI6MTcxNTE2MDUzNywiY3NyZiI
6IjRjOGMzNDU3LTQwMDUtNGM2Yi05NjFjLTBjNTA1OTY3MWUwNiIsImV4cCI6MTcxNTI0Njk
zNywiZmFtaWx5IjoieXRqemR4dG1xd212bHlyayJ9.g0aFkoob7jAwZ5Bv6FzbOPo3_Q6y-
vBpASqXXk19tyw

#HttpOnly_.rubackup.local     TRUE  /     TRUE  0    access_token_cookie

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTcxNTE
```

2MDUzNywianRpIjoiODhmNDI0N2QtNjJlMy00ZjFjLTk4NzgtNDY4OTM2YjRlNTJmIiwidHl
wZSI6ImFjY2VzcyIsInN1YiI6InJ1YmFja3VwIiwibmJmIjoxNzE1MTYwNTM3LCJjc3JmIjo
iNGM4YzM0NTctNDAwNS00YzZiLTk2MWMtMGM1MDU5NjcxZTA2IiwiZXhwIjoxNzE1MTYxNDM
3LCJmYW1pbHkiOiJ5dGp6ZHh0bXF3bXZseXJrIn0.nj-
UbudgyYqGmopRpK9du9gPaZcn_j0_5yhp0A5DLw8

Перевыпуск access_token на основе refresh_token через браузер

Для перевыпуска пары токенов необходимо выполнить следующие действия:

1. Перейдите по адресу https://<app_host>:<app_port>/api/v1/ (рисунок 5):

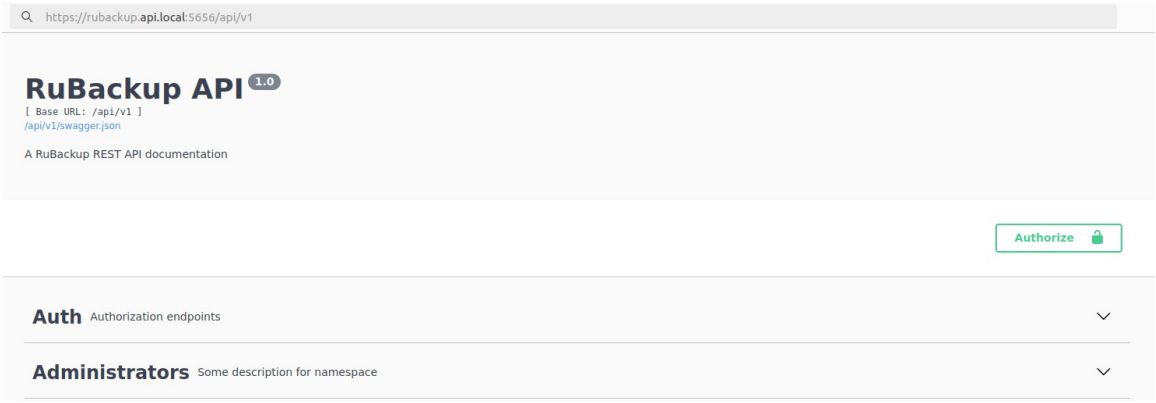


Рисунок 5

2. Перейдите на вкладку «Auth» и выберите эндпоинт «/auth/refresh» (рисунок 6):

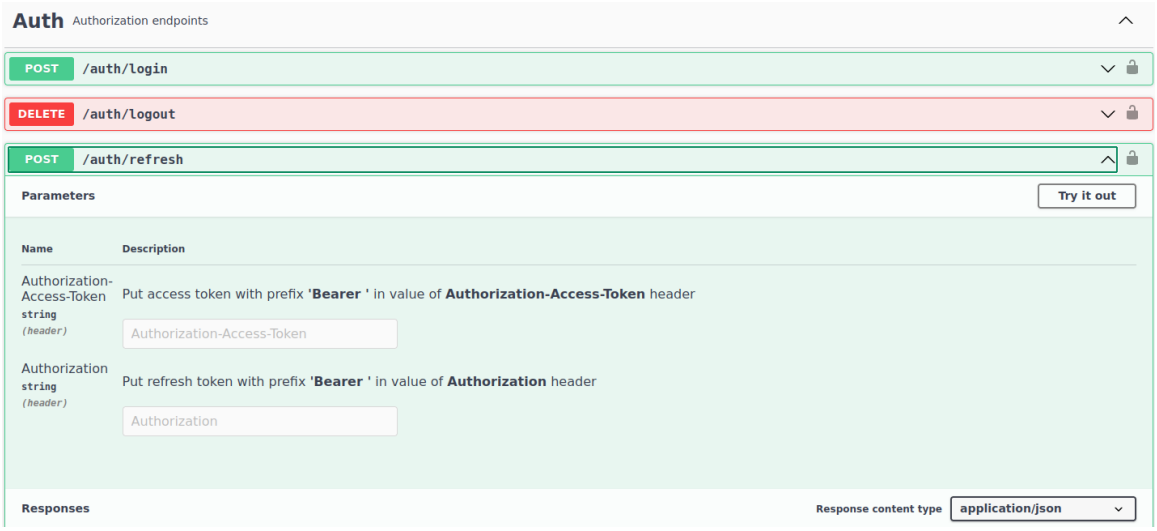


Рисунок 6

3. Нажмите кнопку «Try it out». Если авторизация была пройдена в этом же браузере и access_token и refresh_token все ещё находятся в cookie, то нажмите кнопку «Execute». В ином случае явно укажите токены с префиксом Bearer для параметров Authorization-Access-Token и Authorization (рисунок 7).

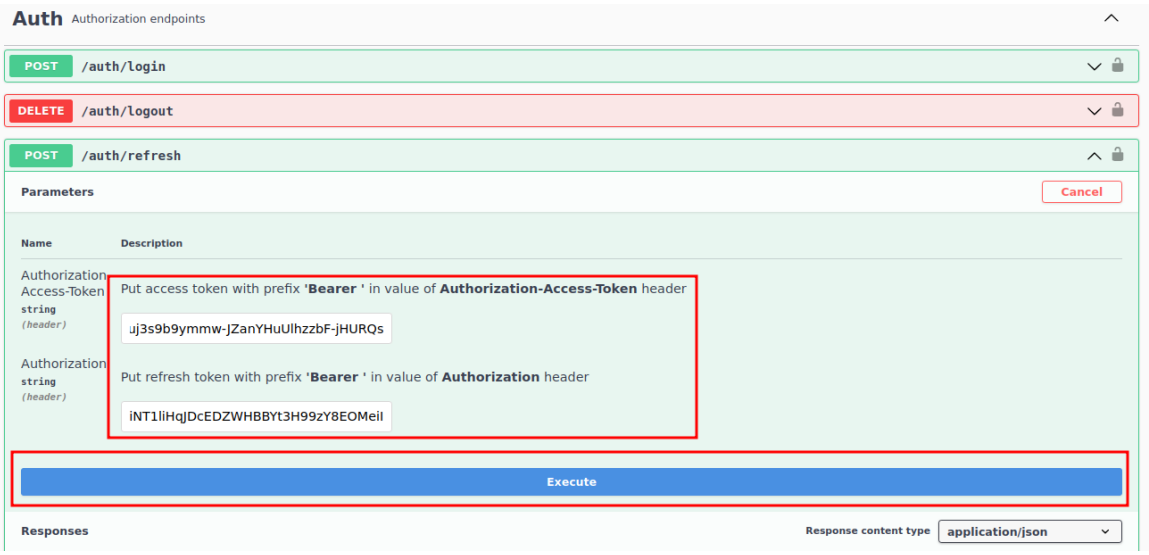


Рисунок 7

В результате проделанных операций Вам вернётся перевыпущенный access_token, перевыпущенные refresh_token и csrf_token, а также сопутствующая информация о пользователе прошедшем авторизацию (рисунок 8). Также сервис автоматически разместит перевыпущенные access_token *и* refresh_token в cookie-файлах.

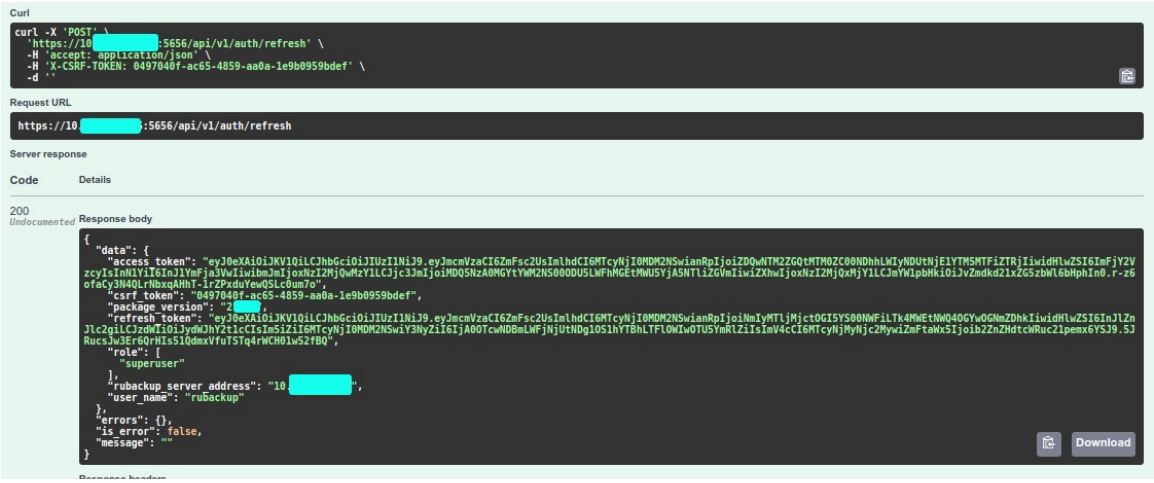


Рисунок 8

Перевыпуск AccessToken на основе RefreshToken через CLI

Чтобы произвести перевыпуск пары токенов через терминал, необходимо отправить POST-запрос с помощью консольной утилиты curl или любым другим удобным способом. В данном примере используется curl:

```
$ curl -k -X POST
'https://api.rubackup.local:5656/api/v1/auth/refresh' \
-H 'accept: application/json' \
-H 'Authorization-Access-Token: Bearer <access_token>' \
 -H 'Authorization: Bearer <refresh_token>' \
-d ''
```

где
https://api.rubackup.local — адрес, где запущен rubackup_api
5656 - порт, который будет использоваться как часть адреса сервера API
/api/v1/auth/refresh — путь до запроса
-H <argument> - значения, передаваемые в Headers
Authorization-Access-Token — значение полученного при авторизации access_token с префиксом Bearer
Authorization — значение полученного при авторизации refresh_token с префиксом Bearer

Результат вернётся в формате JSON:


```
{
  "data": {
"access_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTcxNT
E2MjAwOSwianRpIjoiODhlZDcwYzUtM2Y4Yi00MDVmLWJlMjMtYzcwNTA3YzA5MmViIiwidH
lwZSI6ImFjY2VzcyIsInN1YiI6InJ1YmFja3VwIiwibmJmIjoxNzE1MTYyMDA5LCJjc3JmIj
oiNGM4YzM0NTctNDAwNS00YzZiLTk2MWMtMGM1MDU5NjcxZTA2IiwiZXhwIjoxNzE1MTYyOT
A5LCJmYW1pbHkiOiJleXh1Y250cWVja29ma2p0In0.XHppz9B-eYoEcxZAwcf-
nbXKnu8rC_kXIMRWIU4gZXc",
"csrf_token": "4c8c3457-4005-4c6b-961c-0c5059671e06",
"refresh_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTcxNT
E2MjAwOSwianRpIjoiOGNlYmY4NGMtOTM4MS00YTlhLWE1OGItNmU4ZDM3OTgxMTI0IiwidH
lwZSI6InJlZnJlc2giLCJzdWIiOiJydWJhY2t1cCIsIm5iZiI6MTcxNTE2MjAwOSwiY3NyZi
I6IjRjOGMzNDU3LTQwMDUtNGM2Yi05NjFjLTBjNTA1OTY3MWUwNiIsImV4cCI6MTcxNTI0OD
M3MCwiZmFtaWx5IjoiZXl4dWNudHFlY2tvZmtqdCJ9.H_TPOn-
CF70bYUKa5AP4Ul0MsiPEnzy5foxTZXHi0dE",
    "role": [
      "superuser"
    ],
```

```
    "rubackup_server_address": "10.177.xx.xxx",
    "user_name": "rubackup"
  },
  "errors": {},
  "is_error": false,
  "message": ""
}
```


Если необходимо получить еще и access_token и refresh_token из cookies, то в команде curl следует указать опцию “--cookie-jar -”, например:

```
$ curl -k --cookie-jar - -X POST
'https://api.rubackup.local:5656/api/v1/auth/refresh' \
-H 'accept: application/json' \
-H 'Authorization-Access-Token: Bearer <access_token>' \
 -H 'Authorization: Bearer <refresh_token>' \
-d ''
```

С этой опцией к выводу добавится следующая информация:


```
#HttpOnly_.rubackup.local     TRUE   /     TRUE   0    refresh_token_cookie

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTcxNTE
2MjAwOSwianRpIjoiOGNlYmY4NGMtOTM4MS00YTlhLWE1OGItNmU4ZDM3OTgxMTI0IiwidHl
wZSI6InJlZnJlc2giLCJzdWIiOiJydWJhY2t1cCIsIm5iZiI6MTcxNTE2MjAwOSwiY3NyZiI
6IjRjOGMzNDU3LTQwMDUtNGM2Yi05NjFjLTBjNTA1OTY3MWUwNiIsImV4cCI6MTcxNTI0ODM
3MCwiZmFtaWx5IjoiZXl4dWNudHFlY2tvZmtqdCJ9.H_TPOn-
CF70bYUKa5AP4Ul0MsiPEnzy5foxTZXHi0dE

#HttpOnly_.rubackup.local     TRUE   /     TRUE   0    access_token_cookie

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTcxNTE
2MjAwOSwianRpIjoiODhlZDcwYzUtM2Y4Yi00MDVmLWJlMjMtYzcwNTA3YzA5MmViIiwidHl
wZSI6ImFjY2VzcyIsInN1YiI6InJ1YmFja3VwIiwibmJmIjoxNzE1MTYyMDA5LCJjc3JmIjo
iNGM4YzM0NTctNDAwNS00YzZiLTk2MWMtMGM1MDU5NjcxZTA2IiwiZXhwIjoxNzE1MTYyOTA
5LCJmYW1pbHkiOiJleXh1Y250cWVja29ma2p0In0.XHppz9B-eYoEcxZAwcf-
nbXKnu8rC_kXIMRWIU4gZXc
```


Отзыв токенов AccessToken и RefreshToken через браузер

Для отзыва токенов необходимо выполнить следующие действия:

1. Перейдите по адресу https://<app_host>:<app_port>/api/v1/ (рисунок 9):

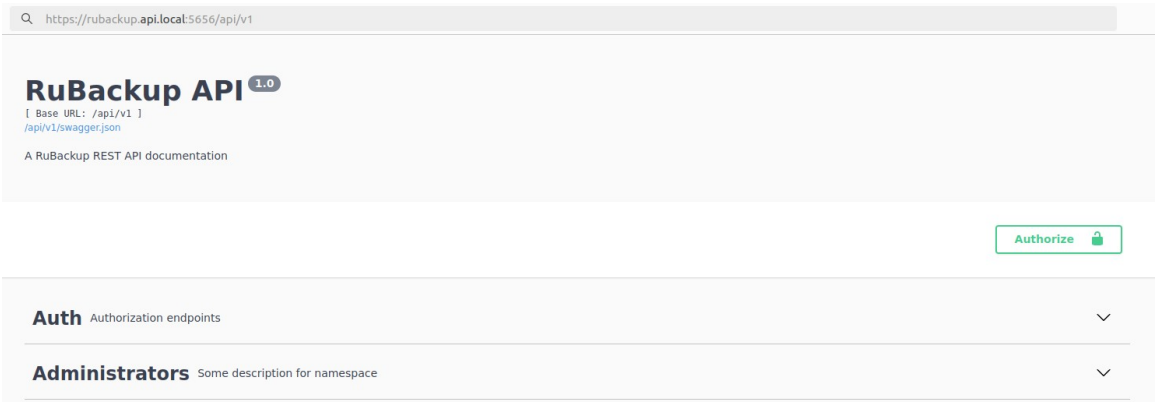


Рисунок 9

2. Перейдите на вкладку «Auth» и выберите эндпоинт «/auth/logout» (рисунок 10). Предварительно убедитесь, что выполнена авторизация (подробнее об авторизации написано в разделе «Авторизация пользователя в веб-интерфейсе»).

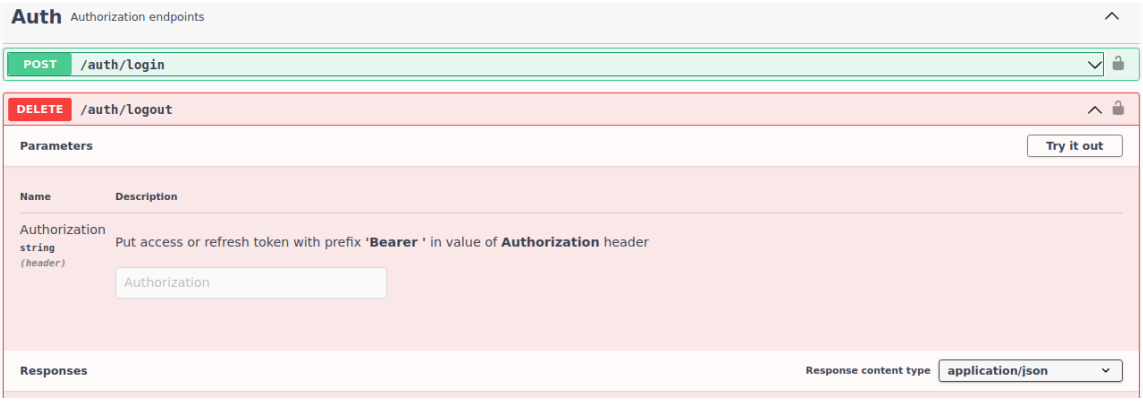


Рисунок 10

3. Нажмите кнопку «Try it out». Если авторизация была пройдена в этом же браузере и access_token и refresh_token все ещё находятся в cookie, то нажмите кнопку «Execute» (рисунок 11). В ином случае явно укажите отзываемый токен для параметра Authorization (подойдет access_token и refresh_token с префиксом Bearer).

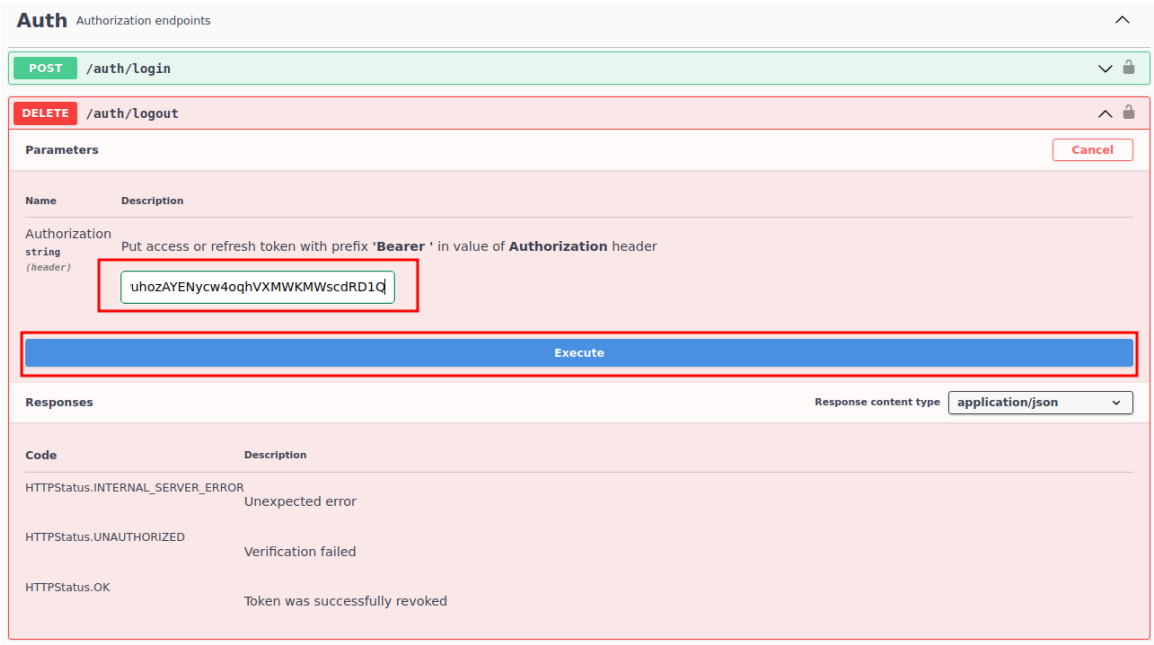


Рисунок 11

4. В результате проделанных операций вернётся информационное сообщение о том, что токены деактивированы (рисунок 12).

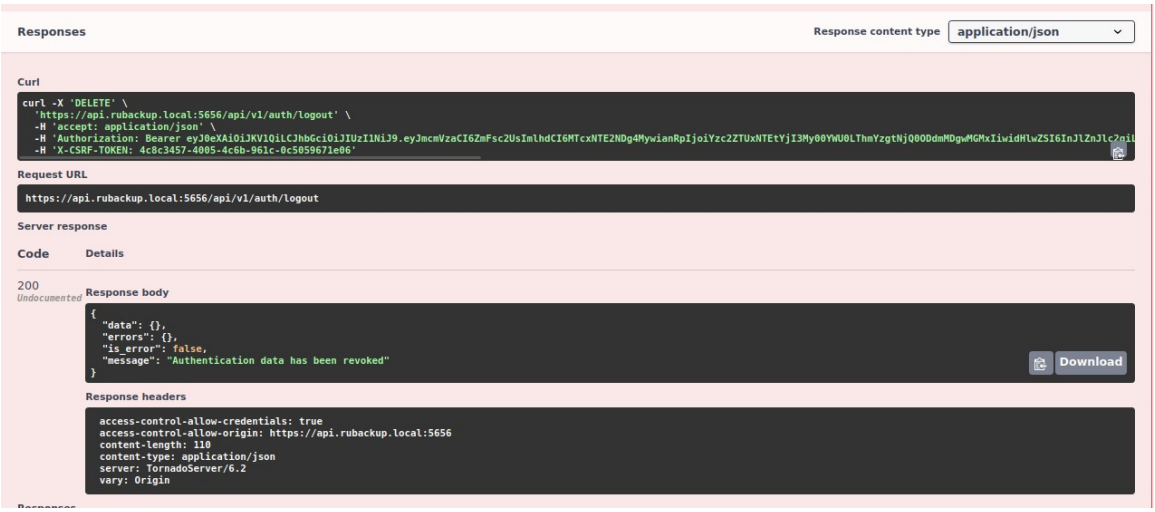


Рисунок 12

Применение полученного csrf_token в Swagger

Для начала работы с ресурсами сервера перейдите по адресу https://<app_host>:<app_port>/api/v1/ в Вашем браузере, выполните получение токенов любым удобным способом и нажмите на кнопку «Authorize» (рисунок 13):

Рисунок 13

В открывшейся форме заполните поле «Value». В данном поле необходимо ввести ранее полученный csrf_token. Затем нажмите кнопку «Authorize» (рисунок 14):

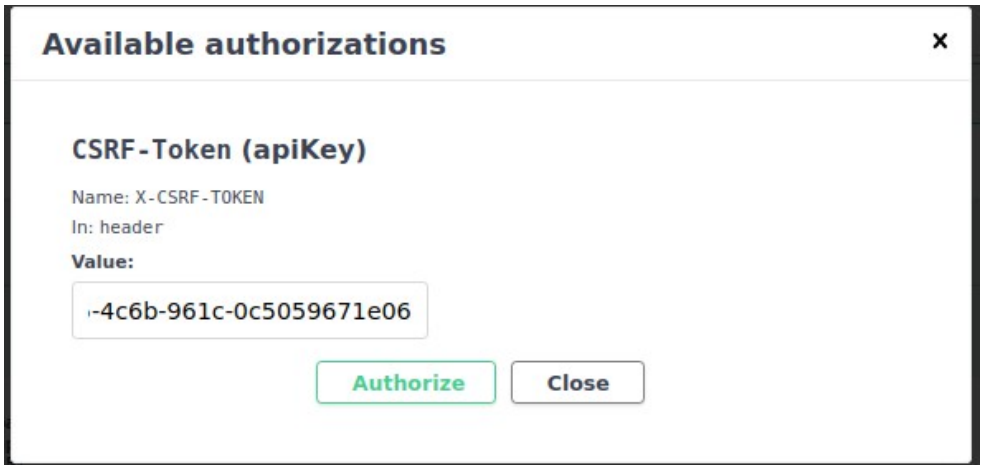


Рисунок 14

Закройте форму нажатием на кнопку «Close» (рисунок 15). Теперь вы авторизованы и можете работать с ресурсами сервера RuBackup.

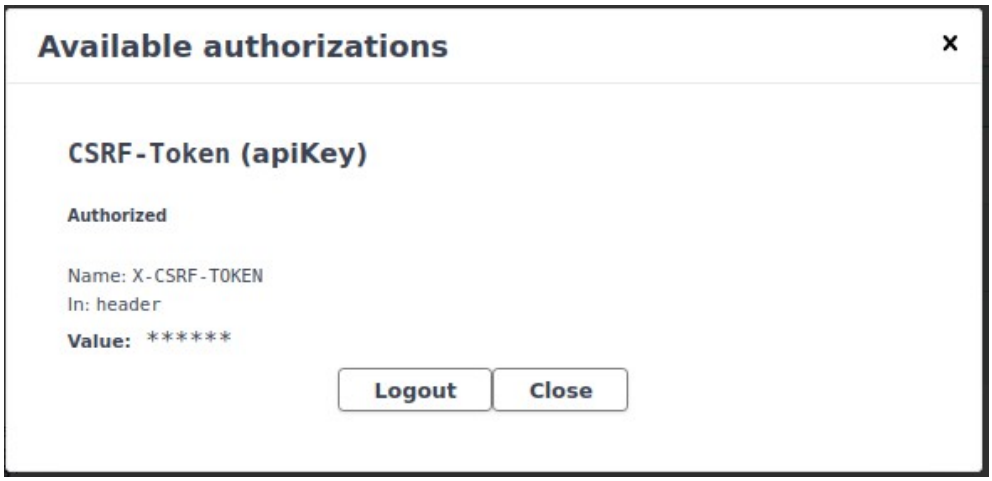


Рисунок 15

Использование CLI для работы с ресурсами

Для того, чтобы отправить запрос на сервер, используя терминал, необходимо отправить запрос с помощью консольной утилиты curl или любым другим удобным способом. В данном примере используется curl. В запросе необходимо передать полученные ранее любым удобным способом csrf-token и access-token.

Для того, чтобы отправить запрос на создание срочной резервной копии файлового ресурса, необходимо выполнить:

```
curl --location 'https://<app_host>:<app_port>/api/v1/task_queue/' \
--header 'Content-Type: application/json' \
--header 'X-CSRF-TOKEN: <полученный при авторизации csrf-token>' \
--header 'Authorization: Bearer <полученный при авторизации access-token>' \
--data '{
  "data": {
    "task": {
      "client": "<client hostname> (<client HW ID>)",
      "resource": "<имя ресурса>",
      "resource_type": "File system",
      "backup_type": "full",
      "pool": "<имя пула>",
      "crypto": "nocrypt",
      "priority": 100,
      "storage_duration": "1 Years",
      "archiving": false
       }
}
}'
```

После отправки данного запроса должен быть получен ответ с информацией о том, что задача добавлена в очередь:

```
{
    "is_error": false,
```

```
    "message": "Resource with name 'Task Queue' was created
successfully! TASK WAS ADDED TO QUEUE:<номер в очереди>",
    "data": {},
    "errors": {}
}
```

Использование браузера для работы с ресурсами

Для отправки запроса на создание срочной резервной копии в браузере необходимо выполнить следующие действия:

1. Перейдите по адресу https://<app_host>:<app_port>/api/v1/ (рисунок 16):

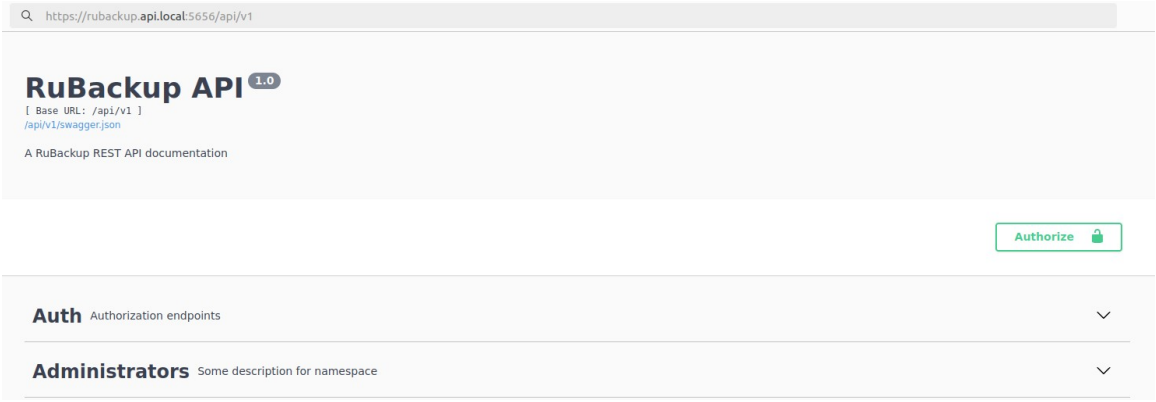


Рисунок 16

2. Авторизуйтесь в Swagger (подробнее об авторизации см. пункт Применение полученного csrf_token в Swagger)

3. Перейдите к разделу Task queue и выберите метод POST (рисунок 17):

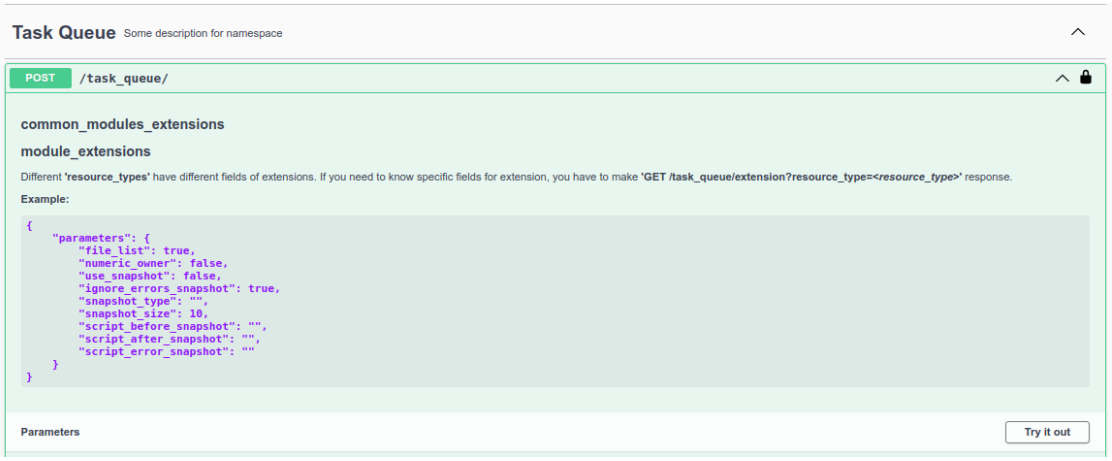


Рисунок 17

4. Нажмите кнопку «Try it out». Заполните обязательные поля тела запроса, например (для создания копии файлового ресурса):

```
{
 "data": {
   "task": {
     "resource_type": "File system",
     "resource": "<имя ресурса>",
     "client": "<client hostname> (<client HW ID>)",
     "backup_type": "full",
     "pool": "<имя пула>",
     "crypto": "nocrypt",
     "priority": 100,
     "storage_duration": "1 Years",
     "archiving": false
   }
 }
}
```

5. Нажмите «Execute» (рисунок 18):

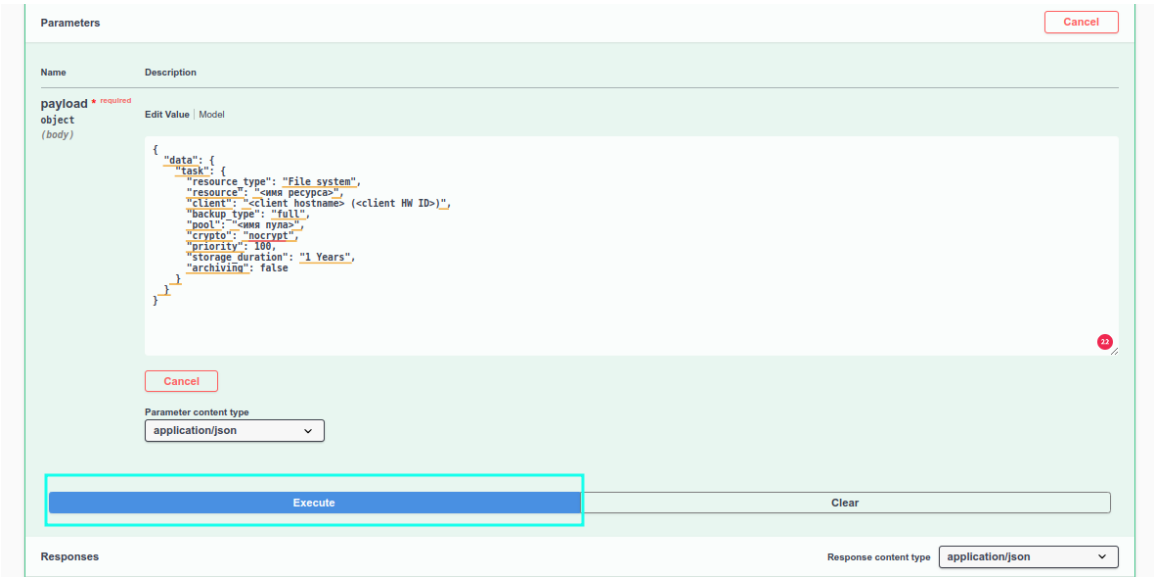


Рисунок 18

6. В результате должен быть получен статус-код 201 CREATED и ответ с информацией о том, что задача добавлена в очередь (рисунок 19):

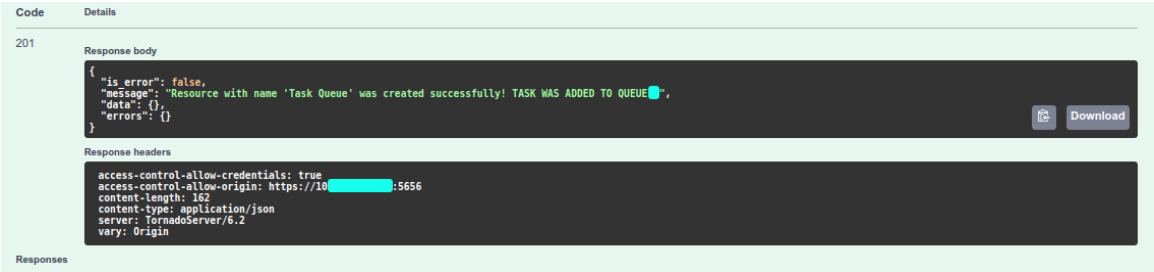


Рисунок 19

Для отслеживания статуса выполнения задачи можно использовать метод GET task_queue.

Доступные для использования ресурсы и их методы

В программном интерфейсе RuBackup перечислены необходимые ресурсы для взаимодействия с сервером резервного копирования (таблица 1). Пользователь может обращаться к каждому представленному ресурсу либо через браузер, *используя спецификацию, которая предоставляется Swagger*, либо через консольную утилиту *curl*.

В каждом из методов каждого эндпоинта должен передаваться CSRF токен.

В части запросов GET в query-параметрах могут передаваться:

- номер страницы списка
- лимит отображения списка (необязательный параметр)
- параметр, по которому будет отсортирован список
- направление сортировки
- фильтр (необязательный параметр). Фильтр указывается в параметр filter_data. Формат, в котором можно указать фильтр (пример, зависит от эндпоинта):
  ➔pool_name="test"
  ➔pool_name~"te"
  ➔last_online<"2024-08-19T19:20:28.614838+03:00"
  ➔last_online>"2024-08-19T19:20:28.614838+03:00"

Таблица 1 – Доступные для использования ресурсы и их методы с описанием их работы

Endpoints	**Methods**	**Статус-коды**	**Тело запроса (пример)** "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	**Описание эндпоинта (функционал)**

Auth	**POST login**	200 OK - успешная авторизация, получены токены
		401 UNAUTHORIZED - Verification failed
			{
			"rubackup_database": "rubackup",
			"rubackup_user": "rubackup",
			"password": "12345"
			}
				Получение токенов и прохождение авторизации. Access токен действителен 15 минут, refresh токен 24 часа, csrf токен - до перезагрузки сервиса rest-api

	DELETE logout	200 OK - успешная деактивация токенов 401 UNAUTHORIZED - Verification failed		Деактивация имеющихся токенов (после выполнения запроса access и refresh токены более не действуют)
	POST refresh	200 OK - успешная деактивация токенов 401 UNAUTHORIZED - Verification failed		Обновление access токена после его истечения (раз в 15 минут)
Administrators	**GET administrators**	200 OK - получен список администраторов		Получение списка администраторов

	DELETE administrators	200 OK - роль администратора удалена
		404 NOT_FOUND - роль администратора не найдена
			{
			"ids": [
			1
			]
			}
				Удаление списка ролей администратор

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	POST administrators	201 CREATED - роль администратора добавлена пользователю	{
			"data": {
			"client_group": "Some group",
			"administrator": "username"
			}
			}
				Назначение пользователю роли администратора

	GET administrators/{id}	200 OK - получена информация об администраторе	Тела запроса нет. В query-параметрах передается id администратора, информацию о котором необходимо получить	Получение информации об одном администраторе
	DELETE administrators/{id}	200 OK - роль администратора удалена 404 NOT_FOUND - роль администратора не найдена	Тела запроса нет. В query-параметрах передается id администратора, которого необходимо удалить	Удаление роли администратора

Backup Strategies	**PATCH backup_strategies**	200 OK - статус стратегии изменен
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - стратегия не найдена
			{
			"data": [
			{
			"strategy_id": 1,
			"status": "wait"
			}
			]
			}
				Изменение статуса стратегии

	GET backup_strategies	200 OK - получен список стратегий		Получение списка стратегий

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	DELETE backup_strategies	200 OK - стратегии удалены
		404 NOT_FOUND - стратегия не найдена
			{
			"ids": [
			1
			]
			}
				Удаление списка стратегий

	POST backup_strategies	201 CREATED - стратегия создана	{
			"data": {
			"name": "my_backup_strategy",
			"status": "wait",
			"pool_id": "Default",
			"crypto": "nocrypt",
			"storage_capacity": 50,
			"description": "some text about backup strategy",
			"validity_start_period": "2024-08-13T14:33:31.359954",
			"validity_end_period": "2025-08-13T14:33:31.360035",
			"verify_flag": true,
			"verify_interval": "1 day",
			"auto_delete_obsoleted_copy_flag": false,
			"inform_when_obsoleted_copy": "Nobody",
			"client_delete_flag": true,
			"full_archive_enabled": false,
			"full_periodic_launch": "1 min",
			"full_storage_duration": "1 day",
			"full_min": 0,
			"full_hour": 0,
			"full_dom": 1,
			"full_mon": 1,
			"full_dow": 1,
			"full_move_copy_flag": false,
			"full_move_copy_while": "1 day",
			"full_move_copy_pool": "Default",
			"inc_archive_enabled": false,
			"inc_periodic_launch": "1 min",
				Создание стратегии

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
			"inc_storage_duration": "1 day", "inc_min": 0, "inc_hour": 0, "inc_dom": 1, "inc_mon": 1, "inc_dow": 1, "inc_move_copy_flag": false, "inc_move_copy_pool": "Default", "inc_move_copy_while": "1 day", "diff_archive_enabled": false, "diff_periodic_launch": "1 min", "diff_storage_duration": "1 day", "diff_min": 0, "diff_hour": 0, "diff_dom": 1, "diff_mon": 1, "diff_dow": 1, "diff_move_copy_flag": false, "diff_move_copy_pool": "Default", "diff_move_copy_while": "1 day", "notify_normal": "Nobody", "notify_normal_cc": "email@domain.ru", "notify_exception": "Nobody", "notify_exception_cc": "email@domain.ru", "notify_verify": "Nobody", "notify_verify_cc": "email@domain.ru", "notify_strategy_validity": "Nobody", "notify_strategy_validity_cc": "email@domain.ru", "notify_end_storage_capacity": "Nobody", "notify_end_storage_capacity_cc": "email@domain.ru" } }	
	PATCH backup_strategies/{id}	200 OK - стратегия	{ "data": {	Изменение параметров

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
		изменена 400 BAD_REQUEST - некорректное тело запроса 404 NOT_FOUND - стратегия не найдена	"name": "my_backup_strategy", "status": "wait", "pool_id": "Default", "crypto": "nocrypt", "storage_capacity": 50, "description": "some text about backup strategy", "validity_start_period": "2024-08-13T14:33:31.359954", "validity_end_period": "2025-08-13T14:33:31.360035", "verify_flag": true, "verify_interval": "1 day", "auto_delete_obsoleted_copy_flag": false, "inform_when_obsoleted_copy": "Nobody", "client_delete_flag": true, "full_archive_enabled": false, "full_periodic_launch": "1 min", "full_storage_duration": "1 day", "full_min": 0, "full_hour": 0, "full_dom": 1, "full_mon": 1, "full_dow": 1, "full_move_copy_flag": false, "full_move_copy_while": "1 day", "full_move_copy_pool": "Default", "inc_archive_enabled": false, "inc_periodic_launch": "1 min", "inc_storage_duration": "1 day", "inc_min": 0, "inc_hour": 0, "inc_dom": 1, "inc_mon": 1, "inc_dow": 1, "inc_move_copy_flag": false, "inc_move_copy_pool": "Default", "inc_move_copy_while": "1 day",	стратегии

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
			"diff_archive_enabled": false, "diff_periodic_launch": "1 min", "diff_storage_duration": "1 day", "diff_min": 0, "diff_hour": 0, "diff_dom": 1, "diff_mon": 1, "diff_dow": 1, "diff_move_copy_flag": false, "diff_move_copy_pool": "Default", "diff_move_copy_while": "1 day", "notify_normal": "Nobody", "notify_normal_cc": "email@domain.ru", "notify_exception": "Nobody", "notify_exception_cc": "email@domain.ru", "notify_verify": "Nobody", "notify_verify_cc": "email@domain.ru", "notify_strategy_validity": "Nobody", "notify_strategy_validity_cc": "email@domain.ru", "notify_end_storage_capacity": "Nobody", "notify_end_storage_capacity_cc": "email@domain.ru" } } Также в query-параметрах передается id стратегии, информацию о которой необходимо изменить	
	GET backup_strategies/{id}	200 OK - получена информация о стратегии	Тела запроса нет. В query-параметрах передается id стратегии, информацию о которой необходимо получить	Получение информации о стратегии

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	DELETE backup_strategies/{id}	200 OK - стратегия удалена 404 NOT_FOUND - стратегия не найдена	Тела запроса нет. В query-параметрах передается id стратегии, которую необходимо удалить	Удалении одной стратегии
Backup Types	**GET backup_types**	200 OK - получен список типов резервного копирования	Тела запроса нет.	Получение списка доступных типов резервного копирования
Client Bandwidth	**GET client _bandwidth**	200 OK - получен список ограничений пропускной способности		Получение списка ограничений пропускной способности клиента

	DELETE client _bandwidth	200 OK - список ограничений пропускной способности удален
		404 NOT_FOUND - ограничение пропускной способности не найдено
			{
			"ids": [
			1
			]
			}
				Удаление списка ограничений пропускной способности клиента

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	POST client _bandwidth	201 CREATED - ограничение пропускной способности создано	{
			"data": {
			"client": "rubackup-client (525a99154f3505a2)",
			"backup_bandwidth": 0,
			"restore_bandwidth": 0,
			"window_start": "14:33:31",
			"window_end": "14:33:31"
			}
			}
				Создание ограничения пропускной способности клиента

	PATCH client _bandwidth/{id}	200 OK - параметры ограничения пропускной способности изменены
		
		400 BAD_REQUEST - некорректное тело запроса
		
		404 NOT_FOUND - ограничение пропускной способности не найдено
			{
			"data": {
			"client": "rubackup-client (525a99154f3505a2)",
			"backup_bandwidth": 0,
			"restore_bandwidth": 0,
			"window_start": "14:33:31",
			"window_end": "14:33:31",
			"id": 1,
			"client_hwid": "525a99154f3505a2"
			}
			}
			
			Также в query-параметрах передается id ограничения, информацию о котором необходимо изменить
				Изменение ограничения пропускной способности клиента

	GET client _bandwidth/{id}	200 OK - получена информация об ограничении пропускной способности	Тела запроса нет. В query-параметрах передается id ограничения, информацию о котором необходимо получить	Получение информации об ограничении пропускной способности клиента

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	DELETE client _bandwidth/{id}	200 OK - ограничение пропускной способности удалено 404 NOT_FOUND - ограничение пропускной способности не найдено	Тела запроса нет. В query-параметрах передается id ограничения, которое необходимо удалить	Удаление ограничения пропускной способности клиента
Client defined storages	**GET client_defined_storages**	200 OK - получен список клиентских хранилищ		Получение списка клиентских хранилищ

	POST client_defined_storages	201 CREATED - клиентское хранилище создано	{
			"data": {
			"pool": "Default",
			"metadata_fs_pool": "Default2",
			"name": "",
			"description": ""
			}
			}
				Добавление клиентского хранилища

	DELETE client_defined_storages	200 OK - список клиентских хранилищ удален
		404 NOT_FOUND - клиентское хранилище не найдено
			{
			"ids": [
			1
			]
			}
				Удаление списка клиентских хранилищ

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	PATCH client_defined_storages/{id}	200 OK - параметры клиентского хранилища изменены
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - клиентское хранилище не найдено
			{
			"data": {
			"name": "",
			"metadata_fs_pool": "Default2",
			"description": ""
			}
			}
			Также в query-параметрах передается id клиентского хранилища, информацию о котором необходимо изменить
				Изменение клиентского хранилища

	GET client_defined_storages/{id}	200 OK - получена информация об клиентском хранилище 404 NOT_FOUND - клиентское хранилище не найдено	Тела запроса нет. В query-параметрах передается id клиентского хранилища, информацию о котором необходимо получить	Получение информации о клиентском хранилище
	DELETE client_defined_storages/{id}	200 OK - клиентское хранилище удалено 404 NOT_FOUND - клиентское хранилище не найдено	Тела запроса нет. В query-параметрах передается id клиентского хранилища, информацию о котором необходимо удалить	Удаление клиентского хранилища
Clients	**GET clients**	200 OK - получен список авторизованных клиентов		Получение списка клиентов

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	DELETE clients	200 OK - список авторизованных клиентов удален
		404 NOT_FOUND - авторизованный клиент не найден
			{
			"ids": [
			1
			]
			}
				Удаление нескольких клиентов

	POST clients/authorize	201 CREATED - клиент авторизован	{
			"data": {
			"clients_ids": [
			1
			]
			}
			}
				Авторизация клиентов

	POST clients/tree	200 OK - получено дерево клиентов
		404 NOT_FOUND - авторизованный клиент не найден
			{
			"data": {
			"clients": [
			{
			"client_group_id": 1,
			"clients_limit": 20,
			"clients_page": 1
			}
			],
			"client_groups_page": 1,
			"client_groups_limit": 20,
			"filter_clients_hostname": "rubackup-client"
			}
			}
				Получение дерева клиентов

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	PATCH clients/{id}	200 OK - параметры клиента изменены
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - авторизованный клиент не найден
			{
			"data": {
			"group_id": "No group",
			"lsf_flag": false,
			"ls_pool": "Default",
			"lrl_flag": false,
			"storage_capacity": 10,
			"description": "Client description",
			"centralized_restore": false,
			"client_side_backup": false,
			"client_side_restore": false
			}
			}
			Также в query-параметрах передается id клиента, информацию о котором необходимо изменить
				Редактирование существующего клиента

	GET clients/{id}	200 OK - получена информация о клиенте	Тела запроса нет. В query-параметрах передается id клиента, информацию о котором необходимо получить	Получение клиента
	DELETE clients/{id}	200 OK - клиент удален (помещен в неавторизованные) 404 NOT_FOUND - авторизованный клиент не найден	Тела запроса нет. В query-параметрах передается id клиента, информацию о котором необходимо получить	Удаление клиента
	GET clients/{id}/resourse_type	200 OK - получена информация о типах ресурса для клиента	Тела запроса нет. В query-параметрах передается id клиента, информацию о типах ресурсов которого котором необходимо получить	Получение доступных типов ресурсов для создания копии

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
Client Groups	**GET client_groups**	200 OK - получена информация о группах клиентов		Получение списка групп клиентов

	DELETE client_groups	200 OK - список групп клиентов удален
		404 NOT_FOUND - группа клиентов не найдена
			{
			"ids": [
			1
			]
			}
				Удаление списка групп клиентов

	POST client_groups	201 CREATED - группа клиентов создана	{
			"data": {
			"name": "group_name",
			"shared": false,
			"clustered": false,
			"description": "some description",
			"launch_attempts_limit": 1
			}
			}
				Добавление групп клиентов

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	PATCH client_groups/{id}	200 OK - параметры группы клиентов изменены
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - группа клиентов не найдена
			{
			"data": {
			"name": "group_name",
			"shared": false,
			"clustered": false,
			"description": "some description",
			"launch_attempts_limit": 1
			}
			}
			Также в query-параметрах передается id группы клиентов, информацию о которой необходимо изменить
				Изменение группы клиентов

	GET client_groups/{id}	200 OK - получена информация о группе клиентов	Тела запроса нет. В query-параметрах передается id группы клиентов, информацию о которой необходимо получить	Получение информации о группе клиентов
	DELETE client_groups/{id}	200 OK - группа клиентов удалена 404 NOT_FOUND - группа клиентов не найдена	Тела запроса нет. В query-параметрах передается id группы клиентов, которую необходимо удалить	Удаление группы клиентов
Clients Log	**GET clients_log**	200 OK - получен список записей в журнале операций клиентов		Получение журнала со списком операций клиентов

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	GET clients_log/{id}	200 OK - получена запись в журнале операций клиентов 404 NOT_FOUND - запись в журнале операций клиентов не найдена	Тела запроса нет. В query-параметрах передается id операции клиента, информацию о которой необходимо получить	Получение одной записи из журнала с операций клиентов
Compression Type	**GET compression_type**	200 OK - получен список типов сжатия		Получение списка доступных типов сжатия
Crypto	**GET crypto**	200 OK - получен список доступных криптоалгоритмов		Получение списка доступных криптоалгоритмов
Deduplication Hash Algorithm	**GET deduplication_hash_algorithm**	200 OK - получен список алгоритмов хеш-функций		Получение списка доступных алгоритмов хеш-функций
Delete Rule Request	**GET delete_rule_requests**	200 OK - получен запросов клиента на удаление правила глобального расписания		Получение списка запросов клиента на удаление правила глобального расписания

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	POST delete_rule_requests/approve	200 NO_CONTENT - запросы клиента на удаление правила глобального расписания одобрены
		
		207 MULTI_STATUS - часть из списка запросов клиента на удаление правила глобального расписания одобрена, часть нет
		
		404 NOT_FOUND - запросы клиента на удаление правила глобального расписания не найдены
			{
			"ids": [
			1
			]
			}
				Одобрение запроса клиента на удаление правила глобального расписания

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	DELETE delete_rule_requests/decline	204 NO_CONTENT - запросы клиента на удаление правила глобального расписания отклонены
		207 MULTI_STATUS - часть из списка запросов клиента на удаление правила глобального расписания отклонена, часть нет
		404 NOT_FOUND - запросы клиента на удаление правила глобального расписания не найдены
			{
			"ids": [
			1
			]
			}
				Отклонение запроса клиента на удаление правила глобального расписания

	GET delete_rule_requests/{id}	200 OK - получена информация о запросе клиента на удаление правила глобального расписания	Тела запроса нет. В query-параметрах передается id запроса, информацию о котором необходимо получить	Получение информации о запросе клиента на удаление правила глобального расписания
Destination Storage Types	**GET destination_storage_types**	200 OK - получен список доступных типов пулов		Получение списка доступных типов пулов

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
Disaster Recovery Plan	**GET disaster_recovery_plan**	200 OK - получен список планов аварийного восстановления		Получение списка планов аварийного восстановления

	DELETE disaster_recovery_plan	200 OK - список планов аварийного восстановления удален	{
			"ids": [
			1
			]
			}
				Удаление списка планов аварийного восстановления

	POST disaster_recovery_plan	201 CREATED - план аварийного восстановления создан	{
			"data": {
			"client": "rubackup-client (525a99154f3505a2)",
			"resource_type": "Aerodisk AIST",
			"resource": "",
			"restore_destination": "",
			"autorun": true,
			"priority": 100,
			"description": ""
			}
			}
				Добавление плана аварийного восстановления

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	PATCH
	disaster_recovery_plan/{id}
		200 OK - параметры плана аварийного восстановления изменены
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - план аварийного восстановления не найден
			{
			"data": {
			"client": "rubackup-client (525a99154f3505a2)",
			"resource_type": "Aerodisk AIST",
			"resource": "",
			"restore_destination": "",
			"autorun": true,
			"priority": 100,
			"description": ""
			}
			}
			Также в query-параметрах передается id плана аварийного восстановления, который необходимо изменить
				Редактирование плана аварийного восстановления

	GET **disaster_recovery_plan/{id}**	200 OK - информация о плане аварийного восстановления получена	Также в query-параметрах передается id плана аварийного восстановления, информацию о котором необходимо получить	Получение информации о плане аварийного восстановления
	DELETE **disaster_recovery_plan/{id}**	200 OK - план аварийного восстановления удален	Также в query-параметрах передается id плана аварийного восстановления, который необходимо удалить	Удаление плана аварийного восстановления

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	POST disaster_recovery_plan/{id}/checking	200 OK - план аварийного восстановления проверен 400 BAD_REQUEST - некорректное тело запроса	Также в query-параметрах передается id плана аварийного восстановления, который необходимо проверить	Проверка плана аварийного восстановления
Disaster Recovery Record Status	**GET disaster_recovery_record_status**			
Dynamic Pool Groups	**GET dynamic_pool_groups**	200 OK - получен список групп пулов		Получение списка динамических группу пулов

	DELETE dynamic_pool_groups	200 OK - группы пулов удалены
		404 NOT_FOUND - группа пулов не найдена
			{
			"ids": [
			1
			]
			}
				Удаление списка динамических групп пулов

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	POST
	dynamic_pool_groups
		201 CREATED - группа пулов создана	{
			"data": {
			"name": "",
			"max_pool_tasks": 10,
			"max_media_server_tasks": 10,
			"max_cpu_usage": 80,
			"max_in_block_operations": 0,
			"max_out_block_operations": 0,
			"calculation_period": 10,
			"description": ""
			}
			}
				Добавление динамической группы пулов

	PATCH
	dynamic_pool_groups/{id}
		200 OK - группа пулов изменена
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - группа пулов не найдена
			{
			"data": {
			"name": "",
			"max_pool_tasks": 10,
			"max_media_server_tasks": 10,
			"max_cpu_usage": 80,
			"max_in_block_operations": 0,
			"max_out_block_operations": 0,
			"calculation_period": 10,
			"description": ""
			}
			}
			Также в query-параметрах передается id группы пулов, информацию о которой необходимо изменить
				Изменение динамической группы пулов

	GET **dynamic_pool_groups/{id}**	200 OK - получена информация о группе пулов	Тела запроса нет. В query-параметрах передается id группы пулов, информацию о которой необходимо получить	Получение информации о динамической группе пулов

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	DELETE dynamic_pool_groups/{id}	200 OK - стратегия удалена 404 NOT_FOUND - группа пулов не найдена	Тела запроса нет. В query-параметрах передается id группы пулов, информацию о которой необходимо удалить	Удаление динамической группы пулов
Dynamic Pool Groups Records	**GET dynamic_pool_groups_records**	200 OK - получен список пулов динамической группы		Получение списка добавленных в динамическую группу пулов

	DELETE dynamic_pool_groups_records	200 OK - список пулов динамической группы удален
		404 NOT_FOUND - пулов динамической группы не найдено
			{
			"ids": [
			1
			]
			}
				Удаление списка добавленных в динамическую группу пулов

	POST dynamic_pool_groups_records	201 CREATED - пул добавлен в группу	{
			"data": {
			"dynamic_pool_group": 1,
			"pool": "Default",
			"enabled": true
			}
			}
				Добавление пула в динамическую группу пулов

	GET dynamic_pool_groups_records/{id}	200 OK - получена информация о пуле динамической группы	Тела запроса нет. В query-параметрах передается id пула из группы пулов, информацию о котором необходимо получить	Получение информации о добавленном в динамическую группу пуле

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	DELETE dynamic_pool_groups_records/{id}	200 OK - пул удален из динамической группы 404 NOT_FOUND - пул в динамической группе не найден	Тела запроса нет. В query-параметрах передается id пула, который необходимо удалить из группы пулов	Удаление добавленного в динамическую группу пула

Global Configuration	**PATCH global_configuration**	200 OK - параметры глобальной конфигурации изменены
		400 BAD_REQUEST - некорректное тело запроса
			{
			"data": {
			"file_transfer_block_size": 16384,
			"data_spred_into_pool": "sequentially",
			"emergency_storage_local_catalog": "/tmp/rubackup_emergency_storage_local_catalog",
			"done_tasks_remove_period": 1440,
			"error_tasks_remove_period": 10080,
			"broken_tasks_remove_period": 10080,
			"killed_tasks_remove_period": 10080,
			"obsoleted_backup_notification_period": 1440,
			"rule_validity_end_notification_period": 1440,
			"strategy_validity_end_notification_period": 1440,
			"rule_storage_capacity_reserve": 0,
			"strategy_storage_capacity_reserve": 0,
			"rule_storage_capacity_notification_period": 1440,
			"strategy_storage_capacity_notification_period": 1440,
			"service_window_start": "00:00:00",
			"service_window_end": "23:59:59",
			"pool_storage_capacity_reserve": 2000000000,
			"tape_library_mount_point": "/opt/rubackup/mnt",
			"suspended_task_restart_period": 1,
			"unload_tape_cartriges_whes_media_server_starts": "yes",
			"ltfs_umount_timeout": 50,
			"max_system_monitoring_records": 3600,
				Редактирование параметров глобальной конфигурации

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
			"monitoring_period": 1, "digital_signature_public_key_obsolence": 1440, "rb_key_hash": "", "verify_archive_after_creation": "no", "kill_task_of_offline_client": "yes", "create_new_task_if_client_offline": "create new task", "auto_delete_archive_from_broken_chain": "no", "consolidating_tape_library_tasks": "yes", "clean_tape_drives_period": 30, "auto_clean_tape_drives": "no", "dedup_clean_unused_blocks": "yes", "dedup_clean_period": 30, "dedup_provide_common_hash_table_for_client": "yes", "dedup_provide_common_hash_table": "yes", "dedup_verify_only_meta_data": "yes", "mandatory_storage_time": 0, "clients_capacity_limits": false, "global_schedule_capacity_limits": false, "backup_strategies_capacity_limits": false, "immutable_archives": true, "delete_archives_with_zeroing": true, "dont_delete_last_gs_rule_archive": true, "dont_delete_last_strategy_rule_archive": true, "wrong_auth_attempts_count": 5, "wrong_auth_block_period": 30, "repository_remove_chain": "yes", "filesystems_clean_period": 30, "show_previous_login_attempts_after_logging_in": "yes", "search_backup_in_cluster_group": true, "bandwith_limit_advantage": "client", "bandwith_client_limit": "minimum", "bandwith_rule_limit": "minimum" }	

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	GET global_configuration	200 OK - параметры глобальной конфигурации получены		Получение параметров глобальной конфигурации

	PATCH global_configuration/service_mode	200 OK - статус сервисного режима изменен
		400 BAD_REQUEST - некорректное тело запроса
			{
			"service_mode": false
			}
				Включение и выключение сервисного режима

Global Schedule	**PATCH global_schedule**	200 OK - статус правила изменен
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - правило глобального расписания не найдено
			{
			"data": [
			{
			"id": 1,
			"status": "wait"
			}
			]
			}
				Изменение статуса правила глобального расписания

	GET global_schedule	200 OK - параметры правила глобального расписания получены		Получение списка правил глобального расписания

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	DELETE global_schedule	200 OK - правило глобального расписания удалено	{
			"ids": [
			1
			]
			}
				Удаление списка правил глобального расписания

	POST global_schedule	201 CREATED - правило глобального расписания создано	{
			"data": {
			"global_schedule": {
			"name": "global_schedule_name",
			"client_id": "rubackup-client (525a99154f3505a2)",
			"resource_type": "File system",
			"resource": "/home/",
			"backup_type": "full",
			"pool_id": "Default",
			"storage_capacity": 10,
			"priority": 100,
			"crypto": "nocrypt",
			"validity_start_period": "2024-09-02T15:40:01.208623",
			"validity_end_period": "2025-09-02T15:40:01.208636",
			"periodic_launch": "1 min",
			"min": 0,
			"hour": 0,
			"dom": 1,
			"mon": 1,
			"dow": 1,
			"verify_flag": true,
			"verify_interval": "1 day",
			"storage_duration": "1 day",
			"move_copy_flag": false,
			"move_copy_pool": "Default",
			"move_copy_while": "1 day",
				Добавление правила глобального расписания

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

			"auto_delete_obsoleted_copy_flag": false,
			"inform_when_obsoleted_copy": "Nobody",
			"client_delete_flag": true,
			"notify_normal": "Nobody",
			"notify_normal_cc": "email@domain.ru",
			"notify_exception": "Nobody",
			"notify_exception_cc": "email@domain.ru",
			"notify_rule_validity": "Nobody",
			"notify_rule_validity_cc": "email@domain.ru",
			"notify_verify": "Nobody",
			"notify_verify_cc": "email@domain.ru",
			"status": "wait",
			"notify_normal_script": "",
			"notify_exception_script": "",
			"notify_end_storage_capacity": "Nobody",
			"notify_end_storage_capacity_cc": "email@domain.ru",
			"restore_script": ""
			},
			"module_extensions": {
			"parameters": {
			"file_list": true,
			"numeric_owner": false,
			"use_snapshot": false,
			"ignore_errors_snapshot": true,
			"snapshot_type": "",
			"snapshot_size": 10,
			"script_before_snapshot": "",
			"script_after_snapshot": "",
			"script_error_snapshot": ""
			}
			},
			"common_modules_extensions": {
			"parameters": {
			"worker_parallelism": 8,
			"enable_multithreading": false,	

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	GET global_schedule/extension	200 OK - информация о дополнительных параметрах получена	Тело запроса отсутствует. В query-параметрах передается название ресурса (resourse_type) или название ресурса и id правила, информацию о дополнительных параметрах которых необходимо получить	

	PATCH global_schedule/{id}	200 OK - параметры правила глобального расписания изменены
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - правило глобального расписания не найдено
			{
			"data": {
			"global_schedule": {
			"name": "global_schedule_name",
			"pool_id": "Default",
			"storage_capacity": 10,
			"priority": 100,
			"validity_start_period": "2024-09-02T15:40:01.208623",
			"validity_end_period": "2025-09-02T15:40:01.208636",
			"periodic_launch": "1 min",
			"min": 0,
			"hour": 0,
			"dom": 1,
			"mon": 1,
			"dow": 1,
			"verify_flag": true,
			"verify_interval": "1 day",
			"storage_duration": "1 day",
			"move_copy_flag": false,
			"move_copy_pool": "Default",
			"move_copy_while": "1 day",
			"auto_delete_obsoleted_copy_flag": false,
			"inform_when_obsoleted_copy": "Nobody",
			"client_delete_flag": true,
			"notify_normal": "Nobody",
			"notify_normal_cc": "email@domain.ru",
			"notify_exception": "Nobody",
				Изменение правила глобального расписания

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

			"notify_exception_cc": "email@domain.ru",
			"notify_rule_validity": "Nobody",
			"notify_rule_validity_cc": "email@domain.ru",
			"notify_verify": "Nobody",
			"notify_verify_cc": "email@domain.ru",
			"status": "wait",
			"notify_normal_script": "",
			"notify_exception_script": "",
			"notify_end_storage_capacity": "Nobody",
			"notify_end_storage_capacity_cc": "email@domain.ru",
			"restore_script": ""
			},
			"module_extensions": {
			"parameters": {
			"file_list": true,
			"numeric_owner": false,
			"use_snapshot": false,
			"ignore_errors_snapshot": true,
			"snapshot_type": "",
			"snapshot_size": 10,
			"script_before_snapshot": "",
			"script_after_snapshot": "",
			"script_error_snapshot": ""
			}
			},
			"common_modules_extensions": {
			"parameters": {
			"worker_parallelism": 8,
			"enable_multithreading": false,
			"enable_flexible_dedup": false,
			"network_parallelism": 8,
			"memory_threshold": 0,
			"deny_memory_exceed": false
			}
			}	

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	GET global_schedule/{id}	200 OK - информация о правиле глобального расписания получена 404 NOT_FOUND - правило глобального расписания не найдено	Тело запроса отсутствует. В query-параметрах передается id правила, информацию о котором необходимо получить	Получение информации о правиле глобального расписания
	DELETE global_schedule/{id}	200 OK - правило глобального расписания удалено 404 NOT_FOUND - правило глобального расписания не найдено	Тело запроса отсутствует. В query-параметрах передается id правила, которое необходимо удалить	Удаление правила глобального расписания
	POST global_schedule/{id}/execution	200 OK - правило глобального расписания выполнено 404 NOT_FOUND - правило глобального расписания не найдено	Тело запроса отсутствует. В query-параметрах передается id правила, которое необходимо выполнить ("принудительно")	
Global Schedule Log	**GET global_schedule_log**	200 OK - получен список записей в журнале правил глобального расписания		Получение списка записей в журнале глобального расписания

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	GET global_schedule_log/{id}	200 OK - получена запись в журнале правил глобального расписания 404 NOT_FOUND - запись в журнале правил глобального расписания не найдена	Тела запроса нет. В query-параметрах передается id операции из журнала правил глобального расписания, информацию о которой необходимо получить	Получение информации о записи в журнале глобального расписания
Library Slots	**GET library_slots**	200 OK - получен список слотов ленточной библиотеки		Получение списка слотов
	GET library_slots/{id}	200 OK - получена информация о слоте ленточной библиотеки	Тела запроса нет. В query-параметрах передается id слота, информацию о которой необходимо получить	Получение информации о слоте

	PATCH library_slots/check_ltfs	200 OK - создана задача на проверку наличия ltfs на картридже
		400 BAD_REQUEST - некорректное тело запроса
			{
			"data": [
			{
			"library_id": 5,
			"volume_tag": "ACS512L9"
			}
			]
			}
				Проверка ltfs на картридже

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	PATCH library_slots/erase_cartridge	200 OK - осоздана задача на стирание картриджа
		400 BAD_REQUEST - некорректное тело запроса
			{
			"data": [
			{
			"library_id": 5,
			"is_long_erase": false,
			"volume_tag": "ACS512L9"
			}
			]
			}
				Стирание картриджа

	PATCH library_slots/export_cartridge	200 OK - осоздана задача на экспорт картриджа
		400 BAD_REQUEST - некорректное тело запроса
			{
			"data": {
			"library_id": 5,
			"volume_tag": "ACS512L9"
			}
			}
				Экспорт картриджа

	PATCH library_slots/import_cartridge	200 OK - импорт картриджа прошел успешно
		400 BAD_REQUEST - некорректное тело запроса
			{
			"data": {
			"library_id": 5,
			"pool_name": "Tape libraries"
			}
			}
				Импорт картриджа

	PATCH library_slots/format_cartridge	200 OK -создана задача на форматирование картриджа
		400 BAD_REQUEST - некорректное тело запроса
			{
			"data": [
			{
			"library_id": 5,
			"volume_tag": "ACS512L9"
			}
			]
			}
				Форматирование картриджа

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	PATCH library_slots/move_cartridge	200 OK - создана задача на перемещение картриджа
		400 BAD_REQUEST - некорректное тело запроса
			{
			"data": {
			"library_id": 5,
			"volume_tag": "ACS512L9",
			"slot_id": 6
			}
			}
				Перемещение картриджа в другой слот

Library Tape Drives	**GET library_tape_drives**	200 OK - получен список приводов ленточной библиотеки		Получение списка ленточных приводов
	GET library_tape_drives/{id}	200 OK - получена информация о приводе ленточной библиотеки	Тела запроса нет. В query-параметрах передается id привода, информацию о которой необходимо получить	Получение информации о ленточном приводе

	POST library_tape_drives	201 CREATED - новый привод добавлен к библиотеке	{
			"data": {
			"ids": [
			4,
			6,
			9
			],
			"tape_library_id": 5
			}
			}
				Добавление привода

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	DELETE library_tape_drives	200 OK - привод ленточной бибилиотеки удален	{
			"ids": [
			2,
			6,
			9
			]
			}
				Удаление списка ленточных приводов

	GET library_tape_drives/candidates	200 OK - информация о доступных для добавления приводах получена	Тела запроса нет. В query-параметрах передается способ добавления привода (ручной или автоматический), hostname медиасервера, на который презентована бибилиотека и sg_device_path робота библиотеки	Получение списка доступных для добавления ленточных приводов

	PATCH library_tape_drives/{id}	200 OK - параметры привода изменены
		
		400 BAD_REQUEST - некорректное тело запроса
			{
			"data": {
			"scsi_path": "[6:0:0:0]",
			"device": "/dev/sg1",
			"sys_transfer_element": 3
			}
			}
			
			Также в query-параметрах передается id привода, информацию о котором необходимо изменить
				Редактирование ленточного привода

	PATCH library_tape_drives/{id}/clean	200 OK - задача на очистку привода создана	Тела запроса нет. В query-параметрах передается id привода, который необходимо очистить	Очистка ленточного привода
License Types	**GET license_types**	200 OK - получен список типов лицензий		Получение списка типов лицензий

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	GET license_types/{id}	200 OK - получена информация о типе лицензии 404 NOT_FOUND - тип лицензии не найден	Тела запроса нет. В query-параметрах передается id типа лицензии, информацию о которой необходимо получить	Получение информации о типе лицензии

Maintainers	**DELETE maintainers**	200 OK - роль сопровождающего удалена
		404 NOT_FOUND - роль сопровождающего не найдена
			{
			"ids": [
			1
			]
			}
				Удаление списку пользователей роли сопровождающего

	GET maintainers	200 OK - получен список сопровождающих		Получение списка сопровождающих

	POST maintainers	201 CREATED - роль администратора добавлена пользователю	{
			"data": {
			"media_server": "hostname",
			"maintainer": "username"
			}
			}
				Назначение пользователю роли сопровождающего

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	DELETE maintainers/{id}	200 OK - роль сопровождающего удалена 404 NOT_FOUND - роль сопровождающего не найдена	Тела запроса нет. В query-параметрах передается id сопровождающего, информацию о котором необходимо удалить	Удаление пользователю роли сопровождающего
	GET maintainers/{id}	200 OK - получена информация осопровождающем	Тела запроса нет. В query-параметрах передается id сопровождающего, информацию о котором необходимо получить	Получение информации о сопровождающем

Media Servers	**DELETE media_servers**	200 OK - список медиасерверов удален
		404 NOT_FOUND - медиасервер не найден
			{
			"ids": [
			1
			]
			}
				Удаление нескольких медиасерверов

	GET media_servers	200 OK - получен список медиасерверов		Получение списка медиасерверов

	POST media_servers/authorize	201 CREATED - медиасервер авторизован	{
			"data": {
			"hostname": "my_hostname"
			}
			}
				Авторизация медиасервера

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	GET media_servers/tree	200 OK - получено дерево медиасерверов		Получение дерева медиасерверов. В списке отображаются медиасерверы, с пулами, медиасерверы, у которых пула. Также в дереве отображаются все пулы медиасерверов и ассоциированные с ними хранилища (директории, блочные устройства, ленточные картриджи) и пулы без ассоциированных устройств
	GET media_servers/filtered_by_tape_library_pool	200 OK - получено дерево медиасерверов		Получение списка медиасерверов, имеющих пул типа ленточная библиотека

	PATCH media_servers/{id}	200 OK - параметры медиасервера изменены
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - медиасервер не найден
			{
			"data": {
			"description": "",
			"local_service_mode": "no"
			}
			}
			Также в query-параметрах передается id медиасервера, информацию о котором необходимо изменить
				Редактирование медиасервера

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	DELETE media_servers/{id}	200 OK - медиасервер удален 404 NOT_FOUND - медиасервер не найден	Тела запроса нет. В query-параметрах передается id медиасервера, который необходимо удалить	Удаление медиасервера
	GET media_servers/{id}	200 OK - получена информация о медиасервере	Тела запроса нет. В query-параметрах передается id медиасервера, информацию о котором необходимо получить	Получение информации о медиасервере
Media Servers Log	**GET media_servers_log**	200 OK - получен список записей в журнале операций медиасерверов		Получение журнала со списком операций медиасерверов
	GET media_servers_log/{id}	200 OK - получена запись в журнале операций медиасерверов 404 NOT_FOUND - запись в журнале операций медиасерверов не найдена	Тела запроса нет. В query-параметрах передается id операции медиасервера, информацию о которой необходимо получить	Получение журнала с операцией медиасервера
Medium	**GET medium_changers**	200 OK - получен список роботов		Получение списка роботов

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
Changers	**GET medium_changers/{id}**	200 OK - получена информация о роботе 404 NOT_FOUND - робот не найден	Тела запроса нет. В query-параметрах передается id робота, информацию о которой необходимо получить	Получение информации о роботе

	PATCH medium_changers/{id}	200 OK - параметры робота изменены
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - робот не найдена
			{
			"data": {
			"scsi_path": "[8:0:0:0]",
			"device": "/dev/sg2"
			}
			}
			Также в query-параметрах передается id робота, информацию о котором необходимо изменить
				Изменение робота

New Rule Requests	**GET new _rule_requests**	200 OK - получен список запросов клиента на добавление правила глобального расписания		Получение списка запросов клиента на добавление правила глобального расписания

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	POST new _rule_requests/approve	200 NO_CONTENT - запросы клиента на добавление правила глобального расписания одобрены
		
		207 MULTI_STATUS - часть из списка запросов клиента на добавление правила глобального расписания одобрена, часть нет
		
		404 NOT_FOUND - запросы клиента на добавление правила глобального расписания не найдены
			{
			"ids": [
			1
			]
			}
				Одобрение запроса клиента на добавление правила глобального расписания

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	DELETE new _rule_requests/decline	200 NO_CONTENT - запросы клиента на добавление правила глобального расписания отклонены
		207 MULTI_STATUS - часть из списка запросов клиента на добавление правила глобального расписания отклонена, часть нет
		404 NOT_FOUND - запросы клиента на добавление правила глобального расписания не найдены
			{
			"ids": [
			1
			]
			}
				Отклонение запроса клиента на добавление правила глобального расписания

	GET new _rule_requests/{id}	200 OK - получена информация о запросе клиента на добавление правила глобального расписания	Тела запроса нет. В query-параметрах передается id запроса, информацию о котором необходимо получить	Получение информации о запросе клиента на добавление правила глобального расписания
Notifications	**GET notifications**	200 OK - получен список уведомлений		Получение списка уведомлений

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	GET notifications/{id}	200 OK - получена информация об уведомлении	Тела запроса нет. В query-параметрах передается id уведомления, информацию о котором необходимо получить	Получение информации о уведомлении
Notification Status	**GET notifications_status**	200 OK - получен список существующих статусов для уведомлений		Получение списка статусов уведомлений
OS Types	**GET os_types**	200 OK - получен список доступных типов операционных систем		Получение списка доступных операционных систем

Pool List	**DELETE pool_list**	200 OK - список пулов удален
		404 NOT_FOUND - пул не найден
			{
			"ids": [
			1
			]
			}
				Удаление списка пулов

	GET pool_list	200 OK - получен список существующих пулов		Получение списка пулов

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	POST pool_list	201 CREATED - пул создан	{
			"data": {
			"pool_name": "my_pool_name",
			"pool_type": "Block device",
			"media_server": "rubackup-server",
			"compression_type": "None",
			"description": "",
			"retention_period": "1 year",
			"ignore_free_space_check": false,
			"algorithm": "streebog",
			"block_size": 131072,
			"hash_length": 256,
			"meta_data_pool": "Default"
			}
			}
				Добавление пула
				
				Набор параметров, которые необходимо передать, зависит от типа пула. В случае, если переданы лишние параметры (которые не подходят создаваемому типу пула), то такие параметры игнорируются.
				
				Параметр compression_type не нужно указывать для пулов типа Block device и Client defined.
				
				Параметры block_size, hash_length, algorithm указываются только для пулов типа Block device.

	GET pool_list/available_for_copy_or_move	200 OK - получен список доступных для перемещения и копирования пулов		Получение списка доступных для перемещения и копирования пулов

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	PATCH pool_list/{id}	200 OK - параметры пула изменены
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - пул не найден
			{
			"data": {
			"pool_name": "my_pool_name",
			"compression_type": "None",
			"description": "",
			"retention_period": "1 year",
			"ignore_free_space_check": false
			}
			}
			Также в query-параметрах передается id пула, информацию о котором необходимо изменить
				Редактирование пула

	DELETE pool_list/{id}	200 OK -пул удален 404 NOT_FOUND - пул не найден	Тела запроса нет. В query-параметрах передается id пула, который необходимо удалить	Удаление пула
	GET pool_list/{id}	200 OK - получена информация о пуле	Тела запроса нет. В query-параметрах передается id пула, информацию о котором необходимо получить	Получение информации о пуле

Pool Substitution	**DELETE pool_substitution**	200 OK - список настроенных подмен пулов удален
		404 NOT_FOUND - подмена пулов не найдена
			{
			"ids": [
			1
			]
			}
				Удаление списка подмены пулов

	GET pool_substitution	200 OK - получен список настроенных подмен пулов		Получение списка подмены пулов

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	POST pool_substitution	201 CREATED - создано правило подмены пулов	{
			"data": {
			"pool": "Default",
			"substitution": "Default2"
			}
			}
				Добавление подмены пулов

	DELETE pool_substitution/{id}	200 OK - правило подмены пулов удалено 404 NOT_FOUND - подмена пулов не найдена	Тела запроса нет. В query-параметрах передается id правила подмены пулов, которое необходимо удалить	Удаление подмены пулов
	GET pool_substitution/{id}	200 OK - получена информация о правило подмены пулов	Тела запроса нет. В query-параметрах передается id правила подмены пулов, информацию о котором необходимо получить	Получение информации о подмене пулов
Repository	**GET repository**	200 OK - получен список сделанных резервных копий		Получение списка объектов репозитория

	DELETE repository	200 OK - список резервных копий удален
		404 NOT_FOUND - резервная копия не найдена
			{
			"ids": [
			1
			]
			}
				Удаление списка резервных копий из репозитория

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	POST repository/ deletion_reference_chain	200 OK - получен список зависимостей для цепочки резервных копий	{
			"ids": [
			1
			]
			}
				Получение списка зависимостей для нескольких цепочек

	GET repository/extension	200 OK - получен список дополнительных параметров восстановления		Получение дополнительных параметров восстановления
	GET repository/{id}	200 OK - получена информация о резервной копии		Получение информации об объекте репозитория
	DELETE repository/{id}	200 OK - резервная копия удалена 404 NOT_FOUND - резервная копия не найдена		Удаление резервной копии из репозитория

	PATCH repository/{id}	200 OK - период хранения резервной копии изменен
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - резервная копия не найдена
			{
			"data": {
			"store_until": "2024-09-07T22:06:09.801038"
			}
			}
				Изменение периода хранения резервной копии

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	POST repository/{id}/copying	201 CREATED - задача на копирование резервной копии создана 404 NOT_FOUND - резервная копия не найдена	Тела запроса нет. В query-параметрах передается id резервной копии и названия пула, в который необходимо провести копирование	Копирование полной резервной копии
	POST repository/{id}/moving	201 CREATED - задача на перемещение резервной копии создана 404 NOT_FOUND - резервная копия не найдена	Тела запроса нет. В query-параметрах передается id резервной копии и названия пула, в который необходимо провести перемещение	Перемещение полной резервной копии

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	POST repository/{id}/restoring	201 CREATED - задача на восстановление резервной копии создана
		
		400 BAD_REQUEST - некорректное тело запроса
		
		404 NOT_FOUND - резервная копия не найдена
			{
			"data": {
			"restore_task": {
			"target_client": "rsafin-primary (55e05458152af100)",
			"destination_path": "/for_restore/",
			"data_deployment": false,
			"granular_restore_files": [
			""
			]
			},
			"module_extensions": {
			"parameters": {
			"new_name": ""
			}
			},
			"common_modules_extensions": {
			"parameters": {
			"worker_parallelism": 8,
			"memory_threshold": 0
			}
			}
			}
			}
			
			Также в query-параметрах передается id резервной копии, которую необходимо восстановить. Для цепочек копий передается id последней разностной копии
				Восстановление резервной копии

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	POST repository/{id}/verification	201 CREATED - задача на верификацию резервной копии создана 404 NOT_FOUND - резервная копия не найдена	Тела запроса нет. В query-параметрах передается id резервной копии, которую необходимо верифицировать	Верификация резервной копии
Repository Log	**GET repository_log**	200 OK - получен список записей в журнале репозитория		Получение списка записей в журнале репозитория
	GET repository_log/{id}	200 OK - получена запись в журнале репозитория 404 NOT_FOUND - запись в журнале репозитория не найдена	Тела запроса нет. В query-параметрах передается id записи в журнале репозитория, информацию о которой необходимо получить	Получение информации о записи в журнале репозитория

Reports	**PATCH reports**	200 OK - статус отчета изменен
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - отчет не найден
			{
			"data": [
			{
			"id": 1,
			"status": "wait"
			}
			]
			}
				Изменение статуса отчета

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	DELETE reports	200 OK - список отчетов удален
		404 NOT_FOUND - отчет не найден
			{
			"ids": [
			1
			]
			}
				Удаление списка отчетов

	GET reports	200 OK - получен список отчетов		Получение списка отчетов

	POST reports	201 CREATED
		- отчет создан
			{
			"data": {
			"name": "my_report_name",
			"owner": "Nobody",
			"min": 0,
			"hour": 0,
			"dom": 1,
			"mon": 1,
			"dow": 1,
			"perpetually": true,
			"start_period": "2024-09-07T22:06:09.809052",
			"end_period": "2025-09-07T22:06:09.809067",
			"status": "wait",
			"notify": "Nobody",
			"notify_cc": "email@domain.ru",
			"description": ""
			}
			}
				Создание отчета

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	PATCH reports/{id}	200 OK - параметры отчета изменены
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - отчет не найден
			{
			"data": {
			"perpetually": true,
			"start_period": "2024-09-07T22:06:09.809052",
			"end_period": "2025-09-07T22:06:09.809067",
			"status": "wait",
			"notify": "Nobody",
			"notify_cc": "email@domain.ru",
			"description": ""
			}
			}
			Также в query-параметрах передается id отчета, информацию о котором необходимо изменить
				Изменение отчета

	DELETE reports/{id}	200 OK - отчет удален 404 NOT_FOUND - отчет не найден	Тела запроса нет. В query-параметрах передается id отчета, который необходимо удалить	Получение информации об отчете
	GET reports/{id}	200 OK - получена информация об отчете 404 NOT_FOUND - отчет не найден	Тела запроса нет. В query-параметрах передается id отчета, информацию о котором необходимо получить	Удаление отчета
Resourse Types	**GET resourse_types**	200 OK - получен список типов ресурсов		Получение списка доступных типов ресурсов

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	GET resourse_types/{id}	200 OK - получена информация о типе ресурса 404 NOT_FOUND - тип ресурса не найден	Тела запроса нет. В query-параметрах передается id типа ресурса, информацию о котором необходимо получить	Получение информации о типе ресурса
Rubackup Server System Monitoring	**GET rubackup_server_system_monitoring**	200 OK - получен список записей в журнале мониторинга сервера		Получение списка записей в журнале мониторинга сервера
	GET rubackup_server_system_monitoring/{id}	200 OK - получена информация о записи в журнале мониторинга сервера 404 NOT_FOUND - запись в журнале мониторинга сервера не найдена	Тела запроса нет. В query-параметрах передается id записи в журнале мониторинга сервера, информацию о которой необходимо получить	Получение информации о записи в журнале мониторинга сервера
Servers HWID	**GET servers_hw_id_tmp**	200 OK - получен список медиасерверов с информацией о них, их HW ID и лицензии		Поиск клиента медиасервера через информацию HW ID сервера

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	GET servers_hw_id_tmp/{id}	200 OK - получена информация о сервере и лицензии 404 NOT_FOUND - информация о медиасервере и лицензии не найдена	Тела запроса нет. В query-параметрах передается id медиасервера, информацию о котором необходимо получить	Получение информации о сервере и лицензии
Server State	**GET server_state**	200 OK - получена информация о текущем состоянии приложения		Получение информации о состоянии приложения. Возвращает список неавторизованных клиентов и медиасерверов, просроченных правил стратегий и глобального расписания, клиентских запросов на добавление и удаление правил
Status Of Rule	**GET status_of_rule**	200 OK - получен список существующих статусов правил		Получение списка доступных статусов правил

Storage Block Devices	**DELETE storage_block_devices**	200 OK - список блочных устройств удален
		404 NOT_FOUND - блочное устройство не найдено
			{
			"ids": [
			1
			]
			}
				Удаление списка блочных устройств

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	GET storage_block_devices	200 OK - получен список блочных устройств		Получение списка блочных устройств

	POST storage_block_devices	201 CREATED - блочное устройство добавлено	{
			"data": {
			"device": "",
			"pool": "Default",
			"overwrite_file_system": false,
			"description": ""
			}
			}
				Добавление блочного устройства

	PATCH storage_block_devices/{id}	200 OK - параметры блочного устройства изменены
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - блочное устройство не найдено
			{
			"data": {
			"description": "",
			"parallel_workers": 2
			}
			}
			Также в query-параметрах передается id блочного устройства, которое необходимо изменить
				Редактирование блочного устройства

	DELETE storage_block_devices/{id}	200 OK - блочное устройство удалено 404 NOT_FOUND - блочное устройство не найдено	Также в query-параметрах передается id блочного устройства, которое необходимо удалить	Удаление блочного устройства

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	GET storage_block_devices/{id}	200 OK - информация о блочном устройстве получена 404 NOT_FOUND - блочное устройство не найдено	Также в query-параметрах передается id блочного устройства, информацию о котором необходимо получить	Получение информации о блочном устройстве

Storage Local Catalogs	**DELETE storage_local_catalogs**	200 OK - список локальных файловых хранилищ удален
		404 NOT_FOUND - локальный файловый хранилище не найдено
			{
			"ids": [
			1
			]
			}
				Удаление нескольких локальных файловых хранилищ

	GET storage_local_catalogs	200 OK - получен список локальных файловых хранилищ		Получение списка локальных файловых хранилищ

	POST storage_local_catalogs	201 CREATED - локальное файловое хранилище добавлено	{
			"data": {
			"path": "",
			"pool": "Default",
			"description": ""
			}
			}
				Добавление локального файлового хранилища

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	PATCH storage_local_catalogs/{id}	200 OK - параметры локального файлового хранилища изменены
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - локальное файловое хранилище не найдено
			{
			"data": {
			"description": ""
			}
			}
			Также в query-параметрах передается id локального файлового хранилища, который необходимо изменить
				Редактирование существующего локального файлового хранилища

	DELETE storage_local_catalogs/{id}	200 OK - блочное устройство удалено 404 NOT_FOUND - локальное файловое хранилище не найдено	Также в query-параметрах передается id локального файлового хранилища, которое необходимо удалить	Удаление локального файлового хранилища
	GET storage_local_catalogs/{id}	200 OK - информация о локальном файловом хранилище получена 404 NOT_FOUND - локальное файловое хранилище не найдено	Также в query-параметрах передается id локального файлового хранилища, информацию о котором необходимо получить	Получение информации о локальном файловом хранилище

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
Strategy Administrators	**GET strategy_administrators**	200 OK - получен список администраторов для стратегии		Получение списка администраторов для стратегии
	GET strategy_administrators/{id}	200 OK - получена информация об администраторе для стратегии 404 NOT_FOUND - информация об администраторе для стратегии не найдена	Тела запроса нет. В query-параметрах передается id администратора стратегии, информацию о котором необходимо получить	Получение информации об администраторе стратегии

	POST strategy_administrators	201 CREATED - администратор для стратегии назначен	{
			"data": {
			"strategy": "Default",
			"administrator": "Nobody"
			}
			}
				Назначение администраторов для стратегии

	DELETE strategy_administrators	200 OK - список администраторов стратегии удален
		404 NOT_FOUND - администратор стратегии не найден
			{
			"ids": [
			1
			]
			}
				Удаление списка администраторов стратегии

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	DELETE strategy_administrators/{id}	200 OK - администратор стратегии удален 404 NOT_FOUND - администратор стратегии не найден	Тела запроса нет. В query-параметрах передается id администратора стратегии, которого необходимо удалить	Удаление администратора стратегии

Strategy Rules	**DELETE strategy_rules**	200 OK - список правил статегии удален
		404 NOT_FOUND - правило стратегии не найдено
			{
			"ids": [
			1
			]
			}
				Удаление списка правил стратегии

	GET strategy_rules	200 OK - получен список правил статегии		Получение списка правил стратегий

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	POST strategy_rules	201 CREATED - правило добавлено в стретегию	{
			"data": {
			"strategy_rule": {
			"client_id": "rubackup-client (525a99154f3505a2)",
			"strategy_id": "Default",
			"resource_type": "File system",
			"resource": "/home/",
			"normal_execution_script": "",
			"exception_execution_script": "",
			"priority": 100,
			"restore_script": ""
			},
			"module_extensions": {
			"parameters": {
			"file_list": true,
			"numeric_owner": false,
			"use_snapshot": false,
			"ignore_errors_snapshot": true,
			"snapshot_type": "",
			"snapshot_size": 10,
			"script_before_snapshot": "",
			"script_after_snapshot": "",
			"script_error_snapshot": ""
			}
			},
			"common_modules_extensions": {
			"parameters": {
			"worker_parallelism": 8,
			"enable_multithreading": false,
			"enable_flexible_dedup": false,
			"network_parallelism": 8,
			"memory_threshold": 0,
			"deny_memory_exceed": false
			}
			}
			}
				Добавление правила в стратегию

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	PATCH strategy_rules/{id}	200 OK - параметры правила стратегии изменены
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - правило стратегии не найдено
			{
			"data": {
			"strategy_rule": {
			"client_id": "rubackup-client (525a99154f3505a2)",
			"strategy_id": "Default",
			"resource_type": "File system",
			"resource": "/home/",
			"normal_execution_script": "",
			"exception_execution_script": "",
			"priority": 100,
			"restore_script": ""
			},
			"module_extensions": {
			"parameters": {
			"file_list": true,
			"numeric_owner": false,
			"use_snapshot": false,
			"ignore_errors_snapshot": true,
			"snapshot_type": "",
			"snapshot_size": 10,
			"script_before_snapshot": "",
			"script_after_snapshot": "",
			"script_error_snapshot": ""
			}
			},
			"common_modules_extensions": {
			"parameters": {
			"worker_parallelism": 8,
			"enable_multithreading": false,
			"enable_flexible_dedup": false,
			"network_parallelism": 8,
			"memory_threshold": 0,
			"deny_memory_exceed": false
			}
			}
			}
				Изменение правила стратегии

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	DELETE strategy_rules/{id}	200 OK - информация о правиле стратегии получена 404 NOT_FOUND - правило стратегии не найдено	Тело запроса отсутствует. В query-параметрах передается id правила, которое необходимо удалить	Удаление правила стратегии
	GET strategy_rules/{id}	200 OK - правило стратегии удалено 404 NOT_FOUND - правило стратегии не найдено	Тело запроса отсутствует. В query-параметрах передается id правила, информацию о котором необходимо получить	Получение информации о правиле стратегии

Supervisors	**DELETE supervisors**	200 OK - роль супервайзера удалена
		404 NOT_FOUND - роль супервайзера не найдена
			{
			"ids": [
			1
			]
			}
				Удаление списка пользователей роли супервайзера

	GET supervisors	200 OK - получен список супервайзеров		Получение списка супервайзеров

	POST supervisors	201 CREATED - роль супервайзера добавлена пользователю	{
			"data": {
			"supervisor": "Nobody"
			}
			}
				Назначение пользователю роли супервайзера

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	DELETE supervisors/{id}	200 OK - получена информация об супервайзере	Тела запроса нет. В query-параметрах передается id супервайзера, информацию о котором необходимо получить	Удаление пользователю роли супервайзера
	GET supervisors/{id}	200 OK - роль супервайзера удалена 404 NOT_FOUND - роль супервайзера не найдена	Тела запроса нет. В query-параметрах передается id супервайзера, которого необходимо удалить	Получение информации о супервайзере

Tape Cartridges	**DELETE tape_cartridges**	200 OK - список картриджей удален
		404 NOT_FOUND - картридж не найден
			{
			"ids": [
			1
			]
			}
				Удаление списка картриджей

	GET tape_cartridges	200 OK - получен список доступных картриджей		Получение списка доступных картриджей

	POST tape_cartridges	201 CREATED - ленточный картридж добавлен	{
			"data": {
			"type": "LTO-9",
			"pool": "tape_pool",
			"volume_tag": "RB10001L9",
			"description": ""
			}
			}
				Добавление ленточного картриджа

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	PATCH tape_cartridges/{id}	200 OK - параметры картриджа изменены
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - картридж не найден
			{
			"data": {
			"volume_tag": "",
			"description": ""
			}
			}
			Также в query-параметрах передается id ленточного картриджа, информацию о которой необходимо изменить
				Редактирование картриджа

	GET tape_cartridges/{id}	200 OK - получена информация о ленточном картридже	Тела запроса нет. В query-параметрах передается id картриджа, информацию о котором необходимо получить	Получение информации о ленточном картридже

Tape Libraries	**DELETE tape_libraries**	200 OK - список ленточных библиотек удален
		404 NOT_FOUND - ленточная библиотека не найдена
			{
			"ids": [
			1
			]
			}
				Удаление списка ленточных библиотек

	GET tape_libraries	200 OK - получен список ленточных библиотек		Получение списка ленточных библиотек

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	POST tape_libraries	201 CREATED - ленточная библиотека добавлена	{
			"data": {
			"pool": "tape_pool",
			"media_server": "rubackup-media",
			"description": "",
			"medium_changer_candidate_id": 1,
			"tape_drives_candidates_ids": [
			1
			]
			}
			}
				Добавление ленточной библиотеки

	DELETE tape_libraries/{id}	200 OK - ленточная библиотека удалена 404 NOT_FOUND - ленточная библиотека не найдена	Тела запроса нет. В query-параметрах передается id ленточной бибилиотеки, информацию о которолй необходимо получить	Удаление ленточной библиотеки
	GET tape_libraries/{id}	200 OK - информация о ленточной библиотеке получена 404 NOT_FOUND - ленточная библиотека не найдена		Получение информации о ленточной библиотеке
	GET tape_libraries/candidates	200 OK - получен список доступных для добавления роботов и приводов	Тела запроса нет. В query-параметрах передается имя медиасервера, которому презентована библиотека, а также способ добаавления	Получение списка доступных для добавления роботов и приводов

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	POST tape_libraries/candidates/info	200 OK - получена информация о роботе и слотах	{
			"data": {
			"media_server": "rubackup-media-server",
			"medium_changer_sg_path": "/dev/sg4"
			}
			}
				Получение информации о роботе и слотах

	POST tape_libraries/tree	200 OK - получено дерево информация о роботе и слотах	{
			"data": {
			"pools": {
			"limit": 50,
			"page": 1
			}
			}
			}
				Получение дерева ленточных библиотек
				
				В списке отображаются:
				1. Ленточные библиотеки, их роботы, их приводы. Список слотов должен быть пустым всегда.
				2. Ленточные пулы, ассоциированные и неассоциированные с библиотекой

	POST tape_libraries/{id}/sync	200 OK - синхронизация библиотеки прошла успешно 422 Unprocessable Entity - синхронизация завершена с ошибкой, бибилиотека содержит незарегистрированный картридж	Тела запроса нет. В query-параметрах передается id библиотеки, информацию о которой необходимо синхронизировать	Синхронизация данных ленточной библиотеки

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
Tape Types	**GET tape_types**	200 OK - получен список доступных типов картриджей		Получение списка доступных типов картриджей
Task Status	**GET task_status**	200 OK - получен список возможных статусов задач		Получение списка возможных статусов задач
Task Log	**GET task_log/{id}**	200 OK - получена запись в журнале задач 404 NOT_FOUND - запись в журнале задач не найдена	Тела запроса нет. В query-параметрах передается id записи в журнале задач, информацию о которой необходимо получить	Получение информации о записи в журнале задач
	GET task_log	200 OK - получен список записей в журнале задач		Получение списка записей в журнале задач

Task Queue	**DELETE task_queue**	200 OK - список задач из очереди задач удален
		404 NOT_FOUND - задача не найдена в очереди
			{
			"ids": [
			1
			]
			}
				Удаление нескольких задач

	GET task_queue	200 OK - получен список задач из очереди задач		Получение очереди задач

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	POST task_queue	201 CREATED - создана задача на срочное PK	{
			"data": {
			"task": {
			"resource_type": "File system",
			"resource": "/home/",
			"client": "rubcp-client (525a99154f3505a2)",
			"backup_type": "full",
			"pool": "Default",
			"crypto": "nocrypt",
			"priority": 100,
			"storage_duration": "1 Years",
			"archiving": false
			},
			"module_extensions": {
			"parameters": {
			"file_list": true,
			"numeric_owner": false,
			"use_snapshot": false,
			"ignore_errors_snapshot": true,
			"snapshot_type": "",
			"snapshot_size": 10,
			"script_before_snapshot": "",
			"script_after_snapshot": "",
			"script_error_snapshot": ""
			}
			},
			"common_modules_extensions": {
			"parameters": {
			"worker_parallelism": 8,
			"enable_multithreading": false,
			"enable_flexible_dedup": false,
			"network_parallelism": 8,
			"memory_threshold": 0,
			"deny_memory_exceed": false
			}
			}
				Создание срочной полной резервной копии

Endpoints	Methods	Статус-коды	**Тело запроса (пример)** "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	**Описание эндпоинта (функционал)**
	GET task_queue/extension	200 OK - информация о дополнительных параметрах получена	Тело запроса отсутствует. В query-параметрах передается название ресурса (resourse_type) или название ресурса и id правила, информацию о дополнительных параметрах которых необходимо получить	Получение дополнительных параметров созданной задачи
	DELETE task_queue/{id}	200 OK - задача удалена из очереди 404 NOT_FOUND - задача не найдена в очереди	Тело запроса отсутствует. В query-параметрах передается id задачи, которую необходимо удалить	Удаление задачи
	GET task_queue/{id}	200 OK - информация о задаче из очереди получена 404 NOT_FOUND - задача не найдена в очереди	Тело запроса отсутствует. В query-параметрах передается id задачи, информацию о которой необходимо получить	Получение информации о задаче из очереди
	DELETE task_queue/{status}	200 OK - задачи с выбранным статусом удалены из очереди 404 NOT_FOUND - задач с выбранным статусом не найдена в очереди	Тело запроса отсутствует. В query-параметрах передается статус, задачи с которым необходимо удалить	Удаление задачи с определенным статусом Возможные статусы: error, killed, obsoleted obsoleted - устаревшие, т.к. в статусе done

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
Tl Task Queue	**GET tl_task_queue**	200 OK - получен список задач из очереди задач ленточных библиотек		Получение списка задач из очереди ленточных библиотек
	GET tl_task_queue/{id}	200 OK - получена информация о задаче из очереди задач ленточных библиотек 404 NOT_FOUND - задача в очереди задач ленточных библиотек не найдена	Тело запроса отсутствует. В query-параметрах передается id задачи из очереди задач ленточных библиотек, информацию о которой необходимо получить	Получение информации о задаче из очереди ленточных библиотек

Unauthorised Clients	**DELETE unauthorised_clients**	200 OK - список неавторизованных клиентов удален
		404 NOT_FOUND - неавторизованные клиенты не найдены
			{
			"ids": [
			1
			]
			}
				Удаление списка неавторизованных клиентов

	GET unauthorised_clients	200 OK - получен список неавторизованных клиентов		Получение списка неавторизованных клиентов

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	DELETE unauthorised_clients/{id}	200 OK - неавторизованный клиент удален 404 NOT_FOUND - неавторизованный клиент не найден	Тело запроса отсутствует. В query-параметрах передается id неавторизованного клиента, которого необходимо удалить	Удаление неавторизованного клиента
	GET unauthorised_clients/{id}	200 OK - получена информация о неавторизованном клиенте	Тело запроса отсутствует. В query-параметрах передается id неавторизованного клиента, информацию о котором необходимо получить	Получение информации о неавторизованном клиенте

Unauthorised Media Servers	**DELETE unauthorised_media_servers**	200 OK - список неавторизованных медиасерверов удален
		404 NOT_FOUND - неавторизованные медиасерверы не найдены
			{
			"ids": [
			1
			]
			}
				Удаление списка неавторизованных медиасерверов

	GET unauthorised_media_servers	200 OK - получен список неавторизованных медиасерверов		Получение списка неавторизованных медиасерверов

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	DELETE unauthorised_media_servers/{id}	200 OK - неавторизованный медиасервер удален 404 NOT_FOUND - неавторизованный медиасервер не найден	Тело запроса отсутствует. В query-параметрах передается id неавторизованного медиасервера, которого необходимо удалить	Удаление неавторизованного медиасервера
	GET unauthorised_media_servers/{id}	200 OK - получена информация о неавторизованном медиасервере	Тело запроса отсутствует. В query-параметрах передается id неавторизованного медиасервера, информацию о котором необходимо получить	Получение информации о неавторизованном медиасервере

User Groups	**DELETE user_groups**	200 OK - список групп пользователей удален
		404 NOT_FOUND - группа пользователей не найдена
			{
			"ids": [
			1
			]
			}
				Удаление списка групп для уведомлений

	GET user_groups	200 OK - список групп пользователей получен		Получение списка групп для уведомлений

	POST user_groups	201 CREATED - создана группа пользователей	{
			"data": {
			"groupname": "my_group_name",
			"description": ""
			}
			}
				Добавление группы уведомлений

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	PATCH user_groups/{id}	200 OK - параметры группы пользователей изменены
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - группа пользователей не найдена
			{
			"data": {
			"groupname": "my_group_name",
			"description": ""
			}
			}
			Также в query-параметрах передается id группы пользователей, информацию о которой необходимо изменить
				Изменение группы для уведомлений

	DELETE user_groups/{id}	200 OK - группа пользователей удалена 404 NOT_FOUND - группа пользователей не найдена	Тела запроса нет. В query-параметрах передается id группы пользователей, которую необходимо удалить	Удаление группы для уведомлений
	GET user_groups/{id}	200 OK - получена информация о группе пользователей 404 NOT_FOUND - группа пользователей не найдена	Тела запроса нет. В query-параметрах передается id группы пользователей, информацию о которой необходимо получить	Получение информации о группе уведомлений

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

Users	**DELETE users**	200 OK - список пользователей удален
		404 NOT_FOUND - пользователь не найден
			{
			"ids": [
			1
			]
			}
				Удаление списка пользователей

	GET users	200 OK - получен список пользователей		Получение списка пользователей

	POST users	201 CREATED - пользователь создан	{
			"data": {
			"group_id": "Nobody",
			"fullname": "",
			"address": "",
			"tel": "",
			"office": "",
			"email": "",
			"username": "",
			"user_password": ""
			}
			}
			Также в query-параметрах передаются:
			• роль пользователя (администратор, супервайзер, сопровождающий)
			• имя группы клиентов (для администраторов)
			• хостнейм медиасервера (для сопровождающих)
				Добавление пользователя

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

	PATCH users/change_password	200 OK - пароль пользователя изменен
		404 NOT_FOUND - пользователь не найден
			{
			"new_password": ""
			}
				Изменение пароля пользователя

	PATCH users/{id}	200 OK - параметры пользователя изменены
		400 BAD_REQUEST - некорректное тело запроса
		404 NOT_FOUND - пользователь не найден
			{
			"data": {
			"group_id": "Nobody",
			"fullname": "",
			"address": "",
			"tel": "",
			"office": "",
			"email": "",
			"new_password": ""
			}
			}
			Также в query-параметрах передается id пользователя, информацию о которой необходимо изменить
				Редактирование пользователей

	DELETE users/{id}	200 OK - пользователь удален 404 NOT_FOUND - пользователь не найден	Тела запроса нет. В query-параметрах передается id пользователя, которого необходимо удалить	Удаление пользователя

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)
	GET users/{id}	200 OK - информация о пользователе получена 404 NOT_FOUND - пользователь не найден	Тела запроса нет. В query-параметрах передается id пользователя, информацию о котором необходимо получить	Получение информации о пользователе
Verification Status	**GET verification_status**	200 OK - получен список доступных статусов верификации		Получение списка доступных статусов верификации
Серверные методы				
Client Task Info	**GET client_task_info**	200 OK - получен журнал клиентских операций по задаче	Тела запроса нет. В query-параметрах передается id задачи из очереди задач, журнал которой необходимо получить	Получение журнала клиентских операций по задаче. Аналогично совершению следующих действий в Менеджере администратора RBM: 1. Перейти на вкладку очереди задач 2. Выбрать задачу в списке 3. Нажать "Журналы" 4. Выбрать "Клиентские журналы"

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

Continue Task Working	**POST continue_task_working**	200 OK - задача возобновлена
		400 BAD_REQUEST - некорректное тело запроса (задачу с таким статусом нельзя возобновить)
			{
			"data": {
			"task_id": 1
			}
			}
				Возобновление приостановленной задачи

Kill Task	**POST kill_task**	200 OK - задача убита
		400 BAD_REQUEST - некорректное тело запроса (задачу с таким статусом нельзя убить)
			{
			"data": {
			"task_id": 1
			}
			}
				Убийство задачи

List Client Block Devices	**POST list_client_block_devices**	200 OK - получен список доступных блочных устройств клиента	{
			"data": {
			"client_hwid": "525a99154f3505a2",
			"path": "/dev/"
			}
			}
				Получение списка доступных блочных устройств указанного клиента

List Client Filesystem	**POST list_client_filesystem**	200 OK - получен список доступных директорий клиента	{
			"data": {
			"client_hwid": "525a99154f3505a2",
			"path": "/"
			}
			}
				Получение списка доступных директорий указанного клиента

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

Module Resource List	**POST module_resource_list**	200 OK - получен список доступных ресурсов модуля	{
			"data": {
			"client_hwid": "525a99154f3505a2",
			"resource_type": "LVM logical volume",
			"secret_method": 0
			}
			}
				Получение списка доступных ресурсов модуля
				
				Например, для PostgreSQL universal это верия postgresql (например, PostgreSQL 16.3)

Pause Task	**POST put_task_on_pause**	200 OK - задача приостановлена
		
		400 BAD_REQUEST - некорректное тело запроса (задачу с таким статусом нельзя приостановить)
			{
			"data": {
			"task_id": 1
			}
			}
				Приостановка выполнения задачи

Repository Record File List	**GET repository_record_file_list**	200 OK - получен список ресурсов, включенных в копию для гранулярного восстановления	Тела запроса нет. В query-параметрах передается id резервной копии, список файлов которой необходимо получить	Получение списка ресурсов, включенных в копию для гранулярного восстановления

Endpoints	Methods	Статус-коды	Тело запроса (пример) "**обязательный ключ**" : "**обязательное значение**" "Необязательный ключ" : "значение"	Описание эндпоинта (функционал)

Restart Task	**POST restart_task**	201 CREATED - задача перезапущена
		400 BAD_REQUEST - некорректное тело запроса (задачу с таким статусом нельзя перезапустить)
			{
			"data": {
			"**tasks_ids": [**
			1
			]
			}
			}
				Перезапуск задачи

Server Hello	**GET server_hello**	200 OK - получена информация о версии rubackup сервера		Получение информации о сервере
Server Task Info	**GET server_task_info**	200 OK - получен журнал серверных операций по задаче	Тела запроса нет. В query-параметрах передается id задачи из очереди задач, журнал которой необходимо получить	Получение журнала серверных операций по задаче. Аналогично совершению следующих действий в Менеджере администратора RBM: 1. Перейти на вкладку очереди задач 2. Выбрать задачу в списке 3. Нажать "Журналы" 4. Выбрать "Серверные журналы"