

RuBackup

Система резервного копирования и восстановления данных

Руководство по установке и взаимодействию с программным интерфейсом RuBackup REST API



RuBackup

Версия 2.1

30.05.2024 г.

Содержание

Введение.....	3
Перед установкой RuBackup REST API.....	4
Базовые требования.....	4
Особенности установки пакетов в Linux.....	5
Установка RuBackup REST API.....	6
Настройка RuBackup REST API.....	6
Настройка журналирования RuBackup REST API.....	8
Настройка запуска RuBackup REST API.....	10
Установка RuBackup REST API на выделенный хост.....	12
Настройка RuBackup REST API на выделенном хосте.....	12
Настройка запуска RuBackup REST API на выделенном хосте.....	14
Использование RuBackup REST API.....	16
Аутентификация пользователя.....	16
Применение полученного CSRF-токена в Swagger.....	26
Доступные для использования ресурсы и их методы.....	28

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 rubackup-common, rubackup-client, rubackup-server и rubackup-rest-api

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

3. Проведен rb_init с конфигурацией типа primary согласно актуальной документации

Подробнее о вариантах установки REST API можно прочитать в разделе “Установка RuBackup REST API”.

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 pr. В настоящем руководстве процедуры установки описаны для пакетного менеджера, который оперирует пакетами deb. Например, процедура 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, такие как 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 rest-api рекомендуется внести изменения в конфигурационный файл postgres для увеличения количества зарезервированных подключений суперпользователя:

например, командой `nano /etc/postgresql/12/main/postgresql.conf`

где 12 — 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. 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. <i>Если он указан некорректно, то при запуске RuBackup API не будут записываться access_token и refresh_token в cookies</i>	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/False

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

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

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

```
GNU nano 6.2      rubackup_api.env
APP_HOST=localhost
APP_PORT=5656
DB_HOST=localhost
DB_PORT=5432
RB_SERVER_HOST=localhost
DEBUG=False
```

Запуск Swagger и Tuscana 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 расплгаются 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 обрабатывать ошибки, куда их записывать.

В нашем случае присутствуют три хэндлера:

- *handler_file* — перехватывает сообщения уровня INFO, применяет определённое форматирование, записывает их в ***/opt/rubackup/log/rubackup_api/access/rubackup-api.log***. Также после полуночи создаётся 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 полуночи создаётся 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]'

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 web-интерфейсом.

Установка RuBackup REST API на выделенный хост

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 rubackup-common, rubackup-client, rubackup-server и rubackup-rest-api
2. Пакеты rubackup-common, rubackup-client, rubackup-server установлены, rb_init проводить 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
```

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. 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. <i>Если он указан некорректно, то при запуске RuBackup API не будут записываться access_token и refresh_token в cookies</i>	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/False

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

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

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

```
GNU nano 4.8 /root/.bashrc
# RuBackup API Settings
export APP_HOST=192.168.10.11
export APP_PORT=5655
export DB_HOST=localhost
export DB_PORT=5432
```

После этого необходимо перезагрузить переменные окружения:

```
# . .bashrc
```

Для запуска Swagger и Tuscany на выделенном хосте в файл переменных окружения 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 Tuscana.

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*.

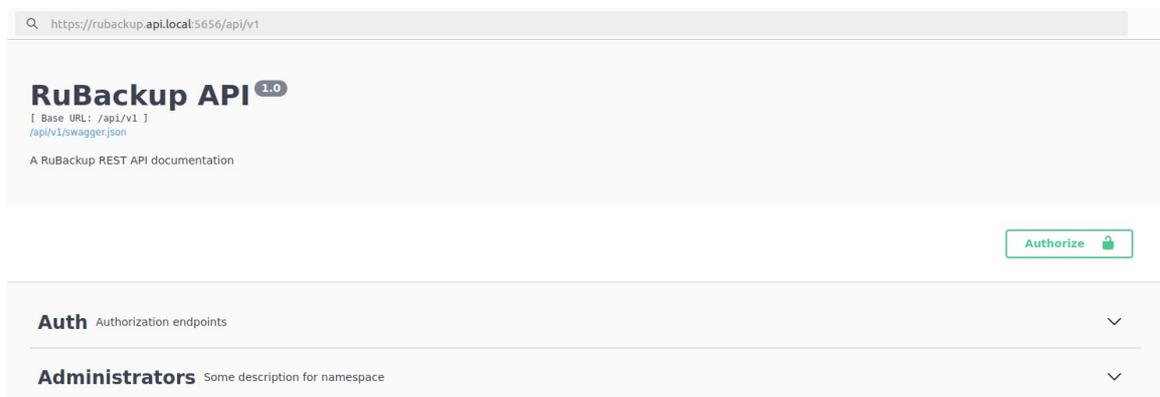
Сгенерированный *access_token* будет действовать в течение 15 минут с момента получения, *refresh_token* — 24 часа с момента получения, *csrf_token* действует до перезагрузки сервиса `rubackup_api`.

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.

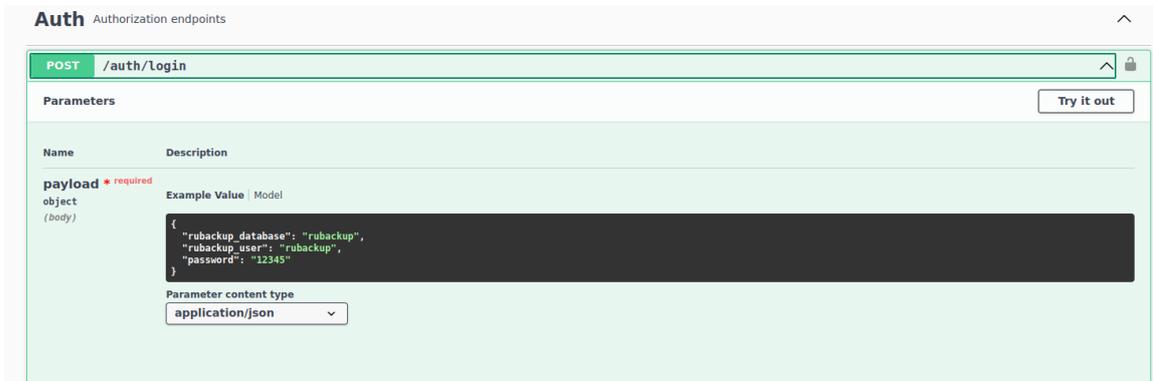
Выпуск AccessToken, RefreshToken и CSRF-Token через браузер

Для получения пары токенов необходимо выполнить следующие действия:

1. Перейдите по адресу https://<app_host>:<app_port>/api/v1/:



2. Перейдите на вкладку «Auth» и выберите эндпоинт “/auth/login”:



The screenshot shows the 'Auth' section of the RuBackup API interface. The endpoint is 'POST /auth/Login'. The 'Parameters' section is expanded to show a 'payload' parameter, which is a required object (body). The 'Example Value' is a JSON object:

```
{  "rubackup_database": "rubackup",  "rubackup_user": "rubackup",  "password": "12345"}
```

. The 'Parameter content type' is set to 'application/json'. A 'Try it out' button is visible in the top right corner.

3. Нажмите кнопку “Try it out”, заполните payload актуальными данными и нажмите “Execute”:



The screenshot shows the 'Auth' section of the RuBackup API interface. The endpoint is 'POST /auth/Login'. The 'Parameters' section is expanded to show a 'payload' parameter, which is a required object (body). The 'Edit Value' is a JSON object:

```
{  "rubackup_database": "rubackup",  "rubackup_user": "rubackup",  "password": "testing12345"}
```

. The 'Parameter content type' is set to 'application/json'. A 'Cancel' button is visible in the top right corner. A large blue 'Execute' button is highlighted with a red box at the bottom of the interface. The 'Responses' section is visible at the bottom, with the 'Response content type' set to 'application/json'.

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

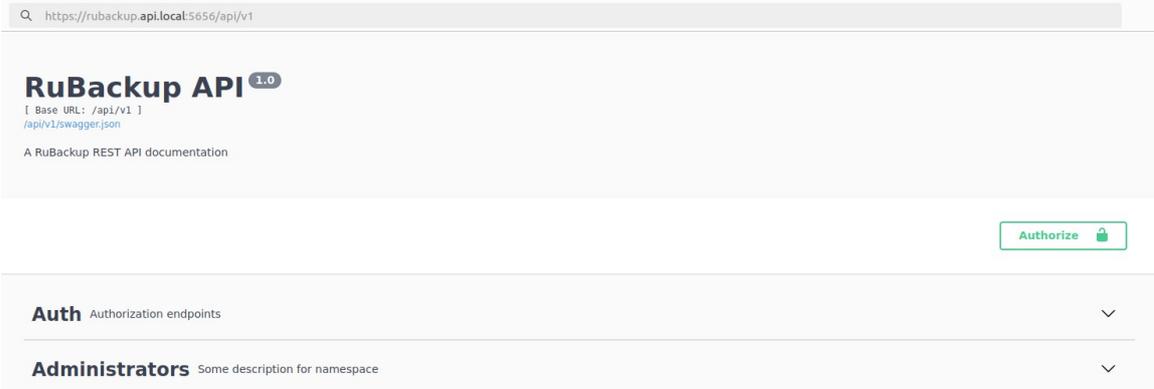

```
#HttpOnly_.rubackup.local TRUE / TRUE 0 access_token_cookie
```

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcmVzaCI6ZmFsc2UsImhhdCI6MTcxNTE2MDUzNywianRpljoiODhmNDI0N2QtNjJlMy00ZjFjLTk4NzgtNDY4OTM2YjRlNTJmIiwidHlwZSI6ImFjY2VzcyIsInN1YiI6ImJ1YmFja3VwIiwibmJmljoiNzE1MTYxwNTM3LCJjc3JmIjoiNGM4YzYzMDNTctNDAwNS00YzZiLTK2MWMtMG1MDU5NjcxZTA2IiwiaWF0IjoiMTYxNDM3LCJmYW1pbHkiOiJ5dGp6ZWh0bXF3bXZseXJrIn0.nj-UbudgyYqGmopRpK9du9gPaZcn_j0_5yhp0A5DLw8
```

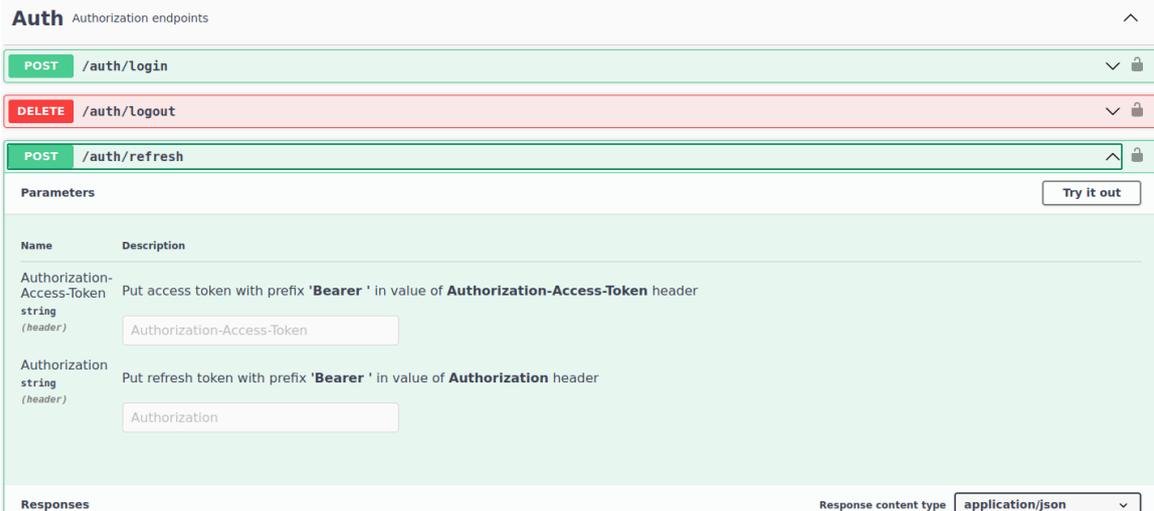
Перевыпуск AccessToken на основе RefreshToken через браузер

Для перевыпуска пары токенов необходимо выполнить следующие действия:

1. Перейдите по адресу https://<app_host>:<app_port>/api/v1/:



2. Перейдите на вкладку «Auth» и выберите эндпоинт “/auth/refresh”:



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

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.eyJmcmVzaCI6ZmFsc2UsImhhdCI6M  
TcxNTE2MjAwOSwianRpljoiODhlZDcwYzUtM2Y4Yi00MDVmLWJIMjMtYzcxNTA  
3YzA5MmViliwidHlwZSI6ImFjY2VzcyIsInN1YiI6ImJ1YmFja3VwliwibmJmljoxNzE1  
MTYyMDA5LCJjc3JmljoiNGM4YzY0NTctNDAwNS00YzZiLTk2MWMtMG1MD  
U5NjcxZTA2liwiZXhwIjoxNzE1MTYyOTA5LCJmYW1pbHkiOiJleXh1Y250cWVja2  
9ma2p0In0.XHppz9B-eYoEcXZAwcf-nbXKnu8rC_kXIMRWIU4gZXc",  
    "csrf_token": "4c8c3457-4005-4c6b-961c-0c5059671e06",  
    "refresh_token":  
    "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcmVzaCI6ZmFsc2UsImhhdCI6M  
TcxNTE2MjAwOSwianRpljoiOGNIYmY4NGMtOTM4MS00YTlhLWE1OGltNmU4Z  
DM3OTgxMTI0liwidHlwZSI6InJlZnJlc2giLCJzdWliOiJydWJhY2t1cCIsIm5iZiI6MTc  
xNTE2MjAwOSwiY3NyZiI6ImFjY2VzcyIsInN1YiI6ImJ1YmFja3VwliwibmJmljoxNzE1  
OTY3MWMUwNiIsImV4cCI6MTcxNTI0ODM3MCwiZmFtaWx5IjoiZXI4dWNudHFiy  
2tvZmtqdCJ9.H_TPOn-CF70bYUKa5AP4UI0MsiPEny5foxTZXHidE",  
    "role": [  
      "superuser"  
    ]  
  },  
}
```


Q https://rubackup.api.local:5656/api/v1

RuBackup API ^{1.0}

[Base URL: /api/v1]
/api/v1/swagger.json

A RuBackup REST API documentation

Authorize 

Auth Authorization endpoints 

Administrators Some description for namespace 

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

Auth Authorization endpoints 

POST /auth/Login  

DELETE /auth/Logout  

Parameters 

Name	Description
Authorization string (header)	Put access or refresh token with prefix ' Bearer ' in value of Authorization header

Authorization

Responses Response content type: application/json 

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 отзываемый токен для параметра Authorization (подойдет access_token и refresh_token с префиксом Bearer).

Применение полученного CSRF-токена 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 1):



Рисунок 1

В открывшейся форме заполните поле «Value». В данном поле необходимо ввести ранее полученный `csrf_token`. Затем нажмите кнопку «Authorize» (рисунок 2):



Рисунок 2

Закройте форму нажатием на кнопку «Close» (рисунок 3). Теперь вы авторизованы и можете работать с ресурсами сервера RuBackup.

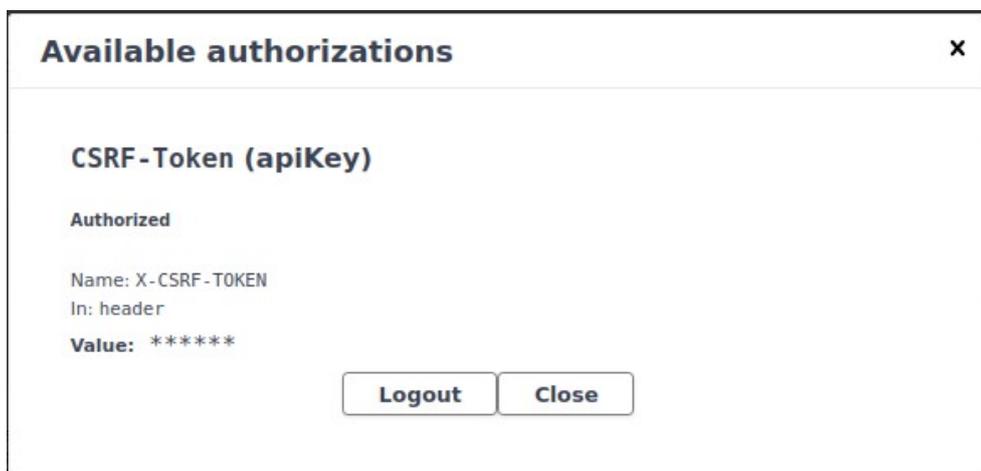


Рисунок 3

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*.

Таблица 1 – Доступные для использования ресурсы и их методы

Название ресурса	Доступные методы
Auth	POST login
	DELETE logout
	POST refresh
Administrators	GET administrators
	DELETE administrators
	POST administrators
	GET administrators/{id}
	DELETE administrators/{id}
Backup Strategies	PATCH backup_strategies
	GET backup_strategies
	DELETE backup_strategies
	POST backup_strategies
	PATCH backup_strategies/{id}
	GET backup_strategies/{id}
	DELETE backup_strategies/{id}
Backup Types	GET backup_types

Название ресурса	Доступные методы
Clients	GET clients
	DELETE clients
	POST clients/authorize
	POST clients/tree
	PATCH clients/{id}
	GET clients/{id}
	DELETE clients/{id}
	GET clients/{id}/resource_type
Client Groups	GET client_groups
	DELETE client_groups
	POST client_groups
	PATCH client_groups/{id}
	GET client_groups/{id}
	DELETE client_groups/{id}
Clients Log	GET clients_log
	GET clients_log/{id}
Compression Type	GET compression_type
Crypto	GET crypto
Deduplication Hash Algorithm	GET deduplication_hash_algorithm
Destination Storage Types	GET destination_storage_types

Название ресурса	Доступные методы
Disaster Recovery Plan	GET disaster_recovery_plan
	DELETE disaster_recovery_plan
	POST disaster_recovery_plan
	PATCH disaster_recovery_plan/{id}
	GET disaster_recovery_plan/{id}
	DELETE disaster_recovery_plan/{id}
	POST disaster_recovery_plan/{id}/checking
Global Configuration	PATCH global_configuration
	PATCH global_configuration/service_mode
	GET global_configuration
Global Schedule	PATCH global_schedule
	GET global_schedule
	DELETE global_schedule
	POST global_schedule
	GET global_schedule/extension
	PATCH global_schedule/{id}
	GET global_schedule/{id}
	DELETE global_schedule/{id}
	POST global_schedule/{id}/execution
Global Schedule Log	GET global_schedule_log
	GET global_schedule_log/{id}

Название ресурса	Доступные методы
License Types	GET license_types
	GET license_types/{id}
Maintainers	DELETE maintainers
	GET maintainers
	POST maintainers
	DELETE maintainers/{id}
	GET maintainers/{id}
Media Servers	DELETE media_servers
	GET media_servers
	POST media_servers/authorize
	GET media_servers/tree
	PATCH media_servers/{id}
	DELETE media_servers/{id}
	GET media_servers/{id}
Media Servers Log	GET media_servers_log
	GET media_servers_log/{id}
Notifications	GET notifications
	GET notifications/{id}
Notification Status	GET notifications_status
OS Types	GET os_types

Название ресурса	Доступные методы
Pool List	DELETE pool_list
	GET pool_list
	POST pool_list
	GET pool_list/available_for_copy_or_move
	PATCH pool_list/{id}
	DELETE pool_list/{id}
	GET pool_list/{id}
Repository	GET repository
	GET repository/extension
	GET repository/{id}
	PATCH repository/{id}
	DELETE repository/{id}
	POST repository/{id}/copying
	POST repository/{id}/moving
	POST repository/{id}/restoring
	POST repository/{id}/verification
Repository Log	GET repository_log
	GET repository_log/{id}
Reports	PATCH reports
	DELETE reports

Название ресурса	Доступные методы
	GET reports
	POST reports
	PATCH reports/{id}
	DELETE reports/{id}
	GET reports/{id}
Resource Types	GET resource_types
	GET resource_types/{id}
Rubackup Server System Monitoring	GET rubackup_server_system_monitoring
	GET rubackup_server_system_monitoring/{id}
Servers HWID	GET servers_hw_id_tmp
	GET servers_hw_id_tmp/{id}
Status Of Rule	GET status_of_rule
Storage Block Devices	DELETE storage_block_devices
	GET storage_block_devices
	POST storage_block_devices
	PATCH storage_block_devices/{id}
	DELETE storage_block_devices/{id}
	GET storage_block_devices/{id}
Storage Local Catalogs	DELETE storage_local_catalogs
	GET storage_local_catalogs

Название ресурса	Доступные методы
	POST storage_local_catalogs
	PATCH storage_local_catalogs/{id}
	DELETE storage_local_catalogs/{id}
	GET storage_local_catalogs/{id}
Strategy Rules	DELETE strategy_rules
	GET strategy_rules
	POST strategy_rules
	GET strategy_rules/extension
	PATCH strategy_rules/{id}
	DELETE strategy_rules/{id}
Supervisors	GET strategy_rules/{id}
	DELETE supervisors
	GET supervisors
	POST supervisors
	DELETE supervisors/{id}
Task Status	GET supervisors/{id}
	GET task_status
Task Log	GET task_log
	GET task_log/{id}
Task Queue	GET task_queue

Название ресурса	Доступные методы
	POST task_queue
	DELETE task_queue
	GET task_queue/extension
	GET task_queue/{id}
	DELETE task_queue/{id}
	DELETE task_queue/{status}
	DELETE unauthorised_clients
Unauthorised Clients	GET unauthorised_clients
	DELETE unauthorised_clients/{id}
	GET unauthorised_clients/{id}
Unauthorised Media Servers	DELETE unauthorised_media_servers
	GET unauthorised_media_servers
	DELETE unauthorised_media_servers/{id}
	GET unauthorised_media_servers/{id}
User Groups	DELETE user_groups
	GET user_groups
	POST user_groups
	PATCH user_groups/{id}
	DELETE user_groups/{id}
	GET user_groups/{id}

Название ресурса	Доступные методы
Users	DELETE users
	GET users
	POST users
	PATCH users/change_password
	PATCH users/{id}
	DELETE users/{id}
	GET users/{id}
Verification Status	GET verification_status
Client Task Info	GET client_task_info
Continue Task Working	POST continue_task_working
Kill Task	POST kill_task
List Client Block Devices	POST list_client_block_devices
List Client Filesystem	POST list_client_filesystem
Pause Task	POST pause_task
Restart Task	POST restart_task
Server Hello	GET server_hello
Server Task Info	GET server_task_info