КИБЕРПРОТЕКТ

КИБЕР
Бэкап Облачный

Версия 26.03

Открытый API

Редакция: 20.04.2026

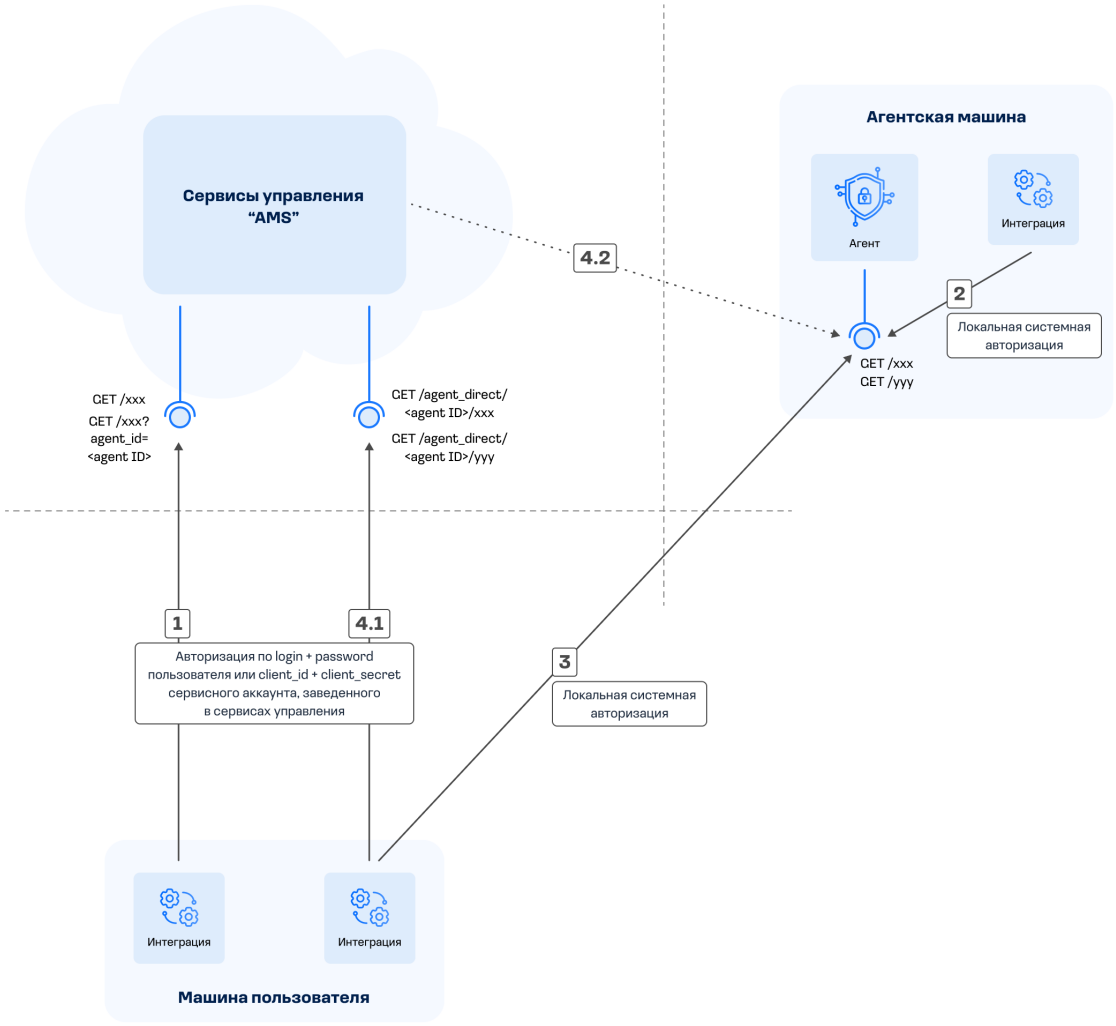
Заявление об авторских правах

Содержание

1 Обзор

1.1 Архитектура

1.1.1 Схема взаимодействия через API



Информационные потоки на схеме:

1 – Запросы открытого API к функциям сервисов управления продукта.

2 – Запросы к API агентов от интеграционных сервисов на агентской машине.

3 – Запросы к API агентов от интеграционных сервисов на машине управления.

4.1 – Запросы открытого API к функциям агентов.

4.2 – Запросы к API агентов от сервисов управления продукта.

1.1.2 Назначение открытого API

Открытый API обеспечивает авторизацию (см. "Авторизация" (стр. 18)), а также доступ к функциям сервисов управления продукта и к функциям агентов, защищающих рабочие нагрузки (виртуальные или физические машины, приложения и т. п. – подробнее см. на странице "Модель лицензирования" (стр. 11)).

Доступ к функциям сервисов управления позволяет управлять следующими ресурсами:

- "Тенант" (стр. 34);
- "Лицензия" (стр. 52);
- "Использование" (стр. 60);
- "Пользователь" (стр. 65);
- "Политика доступа" (стр. 85);
- "Ресурс" (стр. 91);
- "Политика" (стр. 104);
- "Применение политики" (стр. 144);
- "Реквизиты для входа" (стр. 151);
- "Архив резервных копий" (стр. 155);
- "Резервная копия" (стр. 159);
- "Задача" (стр. 162);
- "Действие" (стр. 168).

Доступ к функциям агентов позволяет выполнять следующие действия:

- Резервное копирование
  - Виртуальных и физических машин;
  - Дисков;
  - Файлов;
  - Приложений;
  - Баз данных;
  - Почтовых ящиков;
  - Конфигураций серверов и отдельных виртуальных машин.
- Восстановление из резервных копий.
- Операции с ленточными хранилищами, архивами, резервными копиями.
- Операции с задачами.
- Операции администрирования.

Примечание
Управление функциями агентов запланировано в следующих версиях документации открытого API.

1.2 Модель тенантов

Платформа Кибер Бэкап Облачный использует мультитенантную архитектуру для того, чтобы обеспечить распределенное владение для данных, контроль доступа, лицезирование и настройку для каждой организации и ее физических или логических частей.

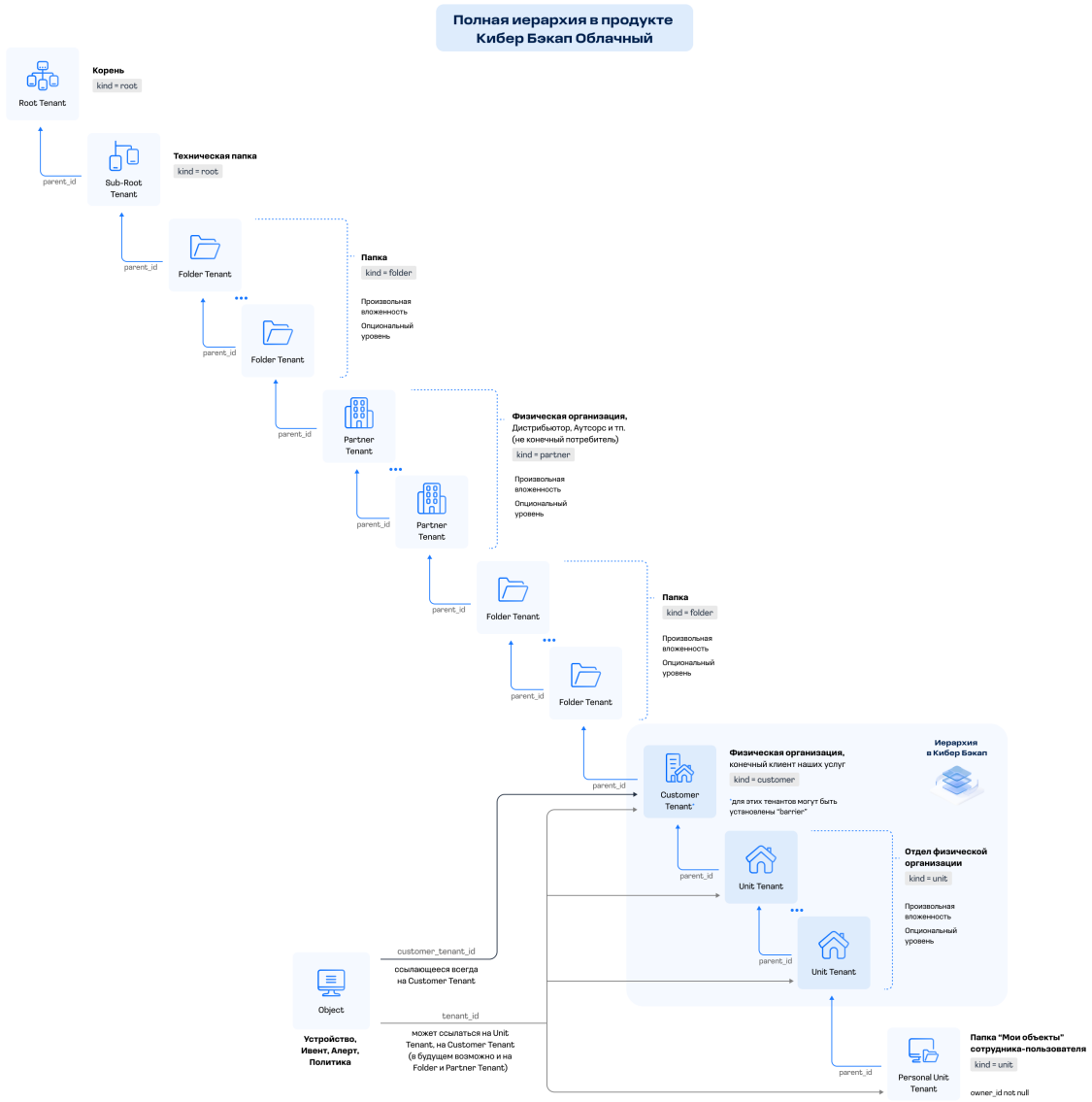
Ниже перечислены типы тенантов, присутствующие в платформе Кибер Бэкап Облачный.

Они связаны отношением родители-дети в следующем порядке:

1. **Partner** (**Партнер**).
   Провайдер сервиса или дистрибьютор, который предоставляет сервис для конечных клиентов. Партнеры управляют субпартнерами и клиентами.
2. **Folder** (**Группа**).
   Логическая группа субпартнеров и клиентов в рамках предоставления услуг провайдером сервиса или дистрибьютором со своими собственными элементами предложения и другим брендом.
3. **Customer** (**Клиент**).
   Клиентская организация, которая приобретает и использует услуги.
4. **Unit** (**Модуль**).
   Логическая единица для управления группами и отделами в рамках клиентской организации.

1.2.1 Пример

Пример иерархии тенантов партнера, группы, клиента, модуля и их соответствующих пользователей.



Полная иерархия применима только для продукта Кибер Бэкап Облачный, для Кибер Бэкап применяется базовая версия структуры, начиная (вниз) с уровня клиентов (customer).

1.3 Модель доступа

1.3.1 Основные правила

Для доступа к платформе Кибер Бэкап Облачный и к тенанту партнеры Киберпротект и клиенты создают личные кабинеты для тех, кто будет их использовать. Для интеграции сервиса партнеры Киберпротект создают служебный личный кабинет, который носит название **Клиент API**.

Для идентификации доступа профилей пользователей и клиентов API в платформе Кибер Бэкап Облачный используются **политики доступа**.

1.3.2 Политики доступа

Политика доступа – это запись, которая определяет:

- кому разрешен доступ (учетная запись пользователя или клиент API);
- к каким ресурсам разрешен доступ (настройки тенанта, рабочие нагрузки, оповещения и т. д.);
- где можно получить доступ (тенант);
- какие действия выполнять (роль).

Например, **партнер** обладает высшим уровнем в иерархии тенантов (см. "Модель тенантов" (стр. 8)) и имеет доступ ко всем субтенантам.

Это означает, что учетная запись партнера обеспечивает ему доступ к каждой настройке субтенантов, конфигурации служб, учетных записей пользователей и т. д.

В то же время учетная запись пользователя не имеет доступа ни к одному соседнему тенанту или к тенантам, расположенным в иерархии выше, или к их данным.

Роль, указанная в политике доступа, определяет, какие действия разрешено выполнять пользователю при работе с тенантами и их данными.

Например, роль **Администратор компании** позволяет выполнять любые действия.

Однако роль **Администратор для чтения** разрешает только просматривать тенанты и их данные.

Примечание
Политики доступа также применяются к клиентам API.

Клиент API наследует политики доступа от учетной записи пользователя, которая его создала.

1.3.3 Режимы управления

В дополнение к обычным правилам платформа Кибер Бэкап Облачный также предоставляет два режима управления, которые могут разрешить доступ к субтенантам для их администраторов в родительских тенантах:

- **Самообслуживание**
  В этом режиме для высокоуровневых администраторов ограничен доступ к этому клиенту: они могут только изменять свойства клиента, но не могут получить доступ к объектам внутри клиента (клиенты, пользователи, службы, резервные копии и другие ресурсы) и управлять ими.
- **Под управлением поставщика услуг**
  В этом режиме клиенту, который используется для поставщика услуг, предоставляется полный доступ: изменение свойств, управление клиентами, пользователями, службами, доступ к резервным копиям и другим ресурсам.

Если администраторы субтенантов хотят предоставить или ограничить доступ администраторам более высокого уровня, они могут включить или выключить доступ поддержки в настройках безопасности своего тенанта.

1.4 Модель лицензирования

Платформа Кибер Бэкап Облачный представляет собой логически изолированный набор функциональности, физически присутствующей в облаке (ЦОД Киберпротект) и в локальной или арендованной инфраструктуре конечного пользователя в виде агента. Одна и та же техническая возможность (например, возможность резервного копирования виртуальных машин) может быть включена в разные модели лицензирования и иметь различные лицензионные условия использования.

Количество потребляемых ресурсов рассчитывается по объёму в ГБ и/или по количеству устройств (виртуальных машин, экземпляров СУБД, почтовых ящиков и т. д.). Количество потребленных сервис-провайдером гигабайт (ГБ) и/или использованных устройств рассчитывается в последний календарный день каждого месяца на основе отчета о потреблении.

1.4.1 Канал распространения

Киберпротект распространяет Кибер Бэкап Облачный через партнерский канал для конечного пользователя. В соответствии с моделью распространения, Киберпротект взаимодействует с дистрибьюторами и провайдерами сервисов и не взаимодействует напрямую с конечными пользователями. Киберпротект разрешает дистрибьюторам регистрировать сервис-провайдера, а тот, в свою очередь, может регистрировать субпровайдера и/или конечных пользователей, используя для этого интерфейсы API или UI.

Выставление счетов работает таким образом: Киберпротект выставляет счета дистрибьюторам, дистрибьюторы – сервис-провайдерам, последние, в свою очередь, – конечным потребителям.

Применяется модель *оплаты по мере использования*.

1.4.2 Функциональность

Функциональность – это отдельная техническая возможность, предоставляемая платформой Кибер Бэкап Облачный или её агентом.

Функциональность может быть предназначена для всего тенанта (**тенантная**) или ориентирована на конкретные рабочие нагрузки (**нагрузочная**).

Тенантные функциональности включаются и отключаются целиком для всего тенанта.

Примеры функциональностей для тенантов:

- Управление ролями пользователей;
- Автоматическое обнаружение рабочих нагрузок из каталога Active Directory;
- Аудит действий пользователя.

Функциональности для конкретных рабочих нагрузок связаны с работой агента и применимы только к конкретным видам рабочих нагрузок.

Примеры функциональностей для рабочих нагрузок:

- Функциональность резервного копирования;
- Функциональность Активной защиты (Active Protection).

1.4.3 Рабочие нагрузки

Рабочая нагрузка – это общий термин для объекта, который защищается, контролируется или управляется платформой Кибер Бэкап Облачный.

Типичные примеры таких защищенных объектов – это пользовательские рабочие станции, виртуальные машины, облачные службы, почтовые ящики, СУБД и т. д.

Примечание
В документации по API рабочие нагрузки также называются *ресурсами*.

1.4.4 Элементы предложения

Элемент предложения представляет собой комбинацию некоторых логически сгруппированных функциональных возможностей, которые могут быть применены к конкретным тенантам и рабочим нагрузкам. Элементы предложения объединяют аналогичные детализированные функциональные возможности, применимые к некоторым аналогичным типам рабочих нагрузок, и являются основным лицензируемым объектом, используемым во всех ключевых сценариях лицензирования.

Элементы предложения позволяют управлять доступными функциями на всех уровнях иерархии модели распространения. Сотрудники компании Киберпротект делают доступными предлагаемые элементы для дистрибьюторов. Дистрибьюторы могут выбирать из доступных предлагаемых элементов и, в свою очередь, делать доступными предлагаемые элементы для сервис-провайдеров, которые используют их для настройки доступных функций для своих конечных клиентов.

Если элемент предложения становится доступен конечному пользователю, его функции становятся доступны для тенанта, профиля пользователя и клиентов API.

Примечание
У конечных пользователей нет своих клиентов, поэтому им не нужно настраивать элементы предложения.

Примеры элементов предложения:

- Виртуальные машины;
- Почтовые ящики;

- Рабочие станции (Advanced Backup);
- Серверы (Advanced Backup).

1.4.5 Квоты

Квота определяет количество элементов предложения, которое возможно использовать.

Используя модель *оплаты по мере использования*, сервис-провайдеры продают элементы предложения своим конечным пользователям, а затем отслеживают и выставляют счета за использование каждого элемента предложения.

Существует два типа квот: "мягкие" и "жесткие".

Квота считается жесткой, если для неё задана величина, на которую квоту можно превысить. При превышении этой величины применяются ограничения на использование сервиса.

Квота считается мягкой, если для неё задана величина превышения "Без ограничений". В таком случае ограничения на использование сервиса не применяются.

1.5 Виды интеграции

1.5.1 Интеграция с сервисами управления, API сервисов управления

Интеграция с сервисами управления обеспечивает авторизацию (см. "Авторизация" (стр. 18)) и доступ к управлению следующими ресурсами:

- "Тенант" (стр. 34);
- "Лицензия" (стр. 52);
- "Использование" (стр. 60) (Потребление);
- "Пользователь" (стр. 65);
- "Политика доступа" (стр. 85);
- "Ресурс" (стр. 91);
- "Политика" (стр. 104);
- "Применение политики" (стр. 144);
- "Реквизиты для входа" (стр. 151);
- "Архив резервных копий" (стр. 155);
- "Резервная копия" (стр. 159);
- "Задача" (стр. 162);
- "Действие" (стр. 168).

Подробнее см. в разделе "Интеграция с сервисами управления" (стр. 15).

1.5.2 Интеграция с агентами

Интеграция с агентами обеспечивает доступ к функциям защиты рабочих нагрузок (виртуальные или физические машины, приложения и т. п. — подробнее см. в разделе "Модель лицензирования" (стр. 11)).

Подробнее об интеграции см. в разделе "Интеграция с агентами" (стр. 173).

2 Интеграция с сервисами управления

2.1 Об API

2.1.1 Варианты установки продукта

Кибер Бэкап может быть установлен в следующих вариантах:

- Кибер Бэкап в облаке вендора;
- Кибер Бэкап в приватном облаке;
- Кибер Бэкап в локальном варианте (on-premise).

2.1.2 Коммуникационный протокол

Для API сервисов управления используется протокол HTTP или HTTPS (в зависимости от вида установки продукта):

- Кибер Бэкап Облачный (в облаке вендора) – всегда используется HTTPS;
- Кибер Бэкап – HTTP или HTTPS, доступно конфигурирование заказчиком.

2.1.3 Стиль API

API сервисов управления построен на стиле REST.

2.1.4 Структура URL API

2.1.4.1 Авторизация

Адрес для авторизации:`(http|https)://<installation address>/idp/token`.

2.1.4.2 Управление ресурсами

`(http|https)://<installation address>/api/<api version>/<rest resource>` – URL-адрес управления ресурсами, в котором:

- installation address – адрес, по которому размещён установленный продукт;
- api version – используемая версия API;
- rest resource – название ресурса.

2.2 Политика версионирования и политика End of life

2.2.1 Версии API

Номер версии API находится в URL: (http|https)://<installation address>/api/**<api version>**/<rest resource>.

Пример

/api/**v1**/

2.2.2 Введение новых версий API

Изменения в API подразделяются на следующие:

- "минорные" – незначительные изменения, которые не приводят к изменению версии API;
- "мажорные" – значительные изменения, меняющие версию API.

2.2.3 Поддержка старых версий, политика End of life (EOL)

API сервисов управления поддерживает 3 последних мажорных версии: 1 актуальная и 2 архивных.

2.3 О документации

Примеры запросов в руководстве по API даны для curl. Примеры с кодом даны на языке Python (версия 3.7), поскольку это простой, понятный и широко используемый язык.

Для чтения примеров требуются базовые знания Python. Вам будет нетрудно понимать примеры кода, даже если вы не слишком хорошо знакомы с Python.

Хороший способ начать работу с Python – это ознакомиться с руководством по Python.

2.3.1 Установка окружения Python

2.3.1.1 Установка окружения Python в Windows

1. Скачайте установщик последней версии Python 3.
2. Запустите установщик.
3. Выберите **Add Python 3.x to PATH**.
4. Щелкните **Install Now**.

   Произойдет установка Python и pip (утилита для локальной установки и настройки пакетов Python) в вашу пользовательскую папку и добавление папок с исполняемыми файлами Python в ваш пользовательский путь.
5. Запустите командную строку.

6. Установите пакет requests, необходимый для отправки запросов к API:

```
pip install requests
```

2.3.1.2 Установка окружения Python в Mac OS X

1. Запустите терминал.
2. Установите Python 3 и pip (утилита для локальной установки и настройки пакетов Python).
3. Установите пакет requests, необходимый для отправки запросов к API:

```
pip install requests
```

2.3.1.3 Установка окружения Python в Linux

1. Запустите терминал.
2. Проверьте, установлен ли у вас уже Python 3:

```
python --version
```

Если Python 3 не установлен, установите его через менеджер пакетов вашего дистрибутива. В зависимости от дистрибутива Linux наименования команд и пакетов, необходимых для установки, отличаются.

- В дистрибутивах Debian и основанных на них, таких как Ubuntu, используйте команду apt:

```
sudo apt-get install python3
```

- В дистрибутивах Red Hat и основанных на них используйте команду yum:

```
sudo yum install python3
```

- В дистрибутивах SUSE и основанных на них используйте команду zypper:

```
sudo zypper install python3
```

3. Установите pip (утилита для локальной установки и настройки пакетов Python).
4. Установите пакет requests, необходимый для отправления запросов к API:

```
pip install requests
```

2.3.2 Запуск оболочки Python и настройка ее сеанса

1. Убедитесь, что настройка окружения Python (см. "Установка окружения Python" (стр. 16)) выполнена корректно.
2. Запустите командную строку.
3. Запустите интерактивную оболочку Python с помощью следующей команды:

```
> python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Теперь вы сможете запускать команды Python и фрагменты кода.

4. Загрузите модули, необходимые для запуска команд в этой инструкции:

```
>>> import requests  # used for sending requests to the API
>>> import json      # used for manipulating JSON data
>>> import pprint    # used for formatting the output of JSON objects received in API responses
```

2.4 Авторизация

Запросы к конечной точке открытого API должны выполняться в соответствии со схемой аутентификации Bearer, если не указано иное. Эта схема требует указания токена в заголовке `Authorization`. Токен – это уникальная зашифрованная строка, генерируемая облачной платформой. Токены устраняют необходимость в передаче учетных данных пользователя с запросами.

Токен выдается конечной точкой /idp/token системой управления учетными записями открытого API.

Чтобы сгенерировать токен, ознакомьтесь с разделом "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19).

Пример

Запрос, отправленный через curl для API, имеет следующую форму:

```
curl -X POST -s <installation address>/api/<api version>/<endpoint> \
   --header "Authorization: Bearer
8770b34b74f9e4d9424eff50c38182bb4ae7f5596582ae61900b1b6a23e3ec58"
```

Если, например, срок действия токена, указанного в заголовке Authorization, истек или ваша учетная запись отключена, API вернет код состояния 401 (см. "Коды состояния HTTP" (стр. 27)).

Внимание
По соображениям безопасности время истечения срока действия токена установлено равным двум часам. По истечении этого времени API вернет код состояния 401 (см. "Коды состояния HTTP" (стр. 27)).

2.4.1 Аутентификация на платформе Кибер Бэкап Облачный

Внимание
Для аутентификации на платформе Кибер Бэкап Облачный у вас должна быть активирована учетная запись администратора.

Пройдя аутентификацию на платформе Кибер Бэкап Облачный с помощью токена доступа, интеграция получит право выполнять операции в тенанте и его субтенантах от имени администратора.

Примечание
Перед началом работы вы должны запустить оболочку Python и настроить ее сеанс (см. "Запуск оболочки Python и настройка ее сеанса" (стр. 17)).

Модули **requests**, **json** и **pprint** должны быть загружены в интерактивную оболочку.

2.4.1.1 Выполнение аутентификации на платформе Кибер Бэкап Облачный

Регистрация клиента API в тенанте

1. Следуйте процедурам, описанным в разделе Управление клиентами API Руководства администратора партнера Кибер Бэкап Облачный.
2. Задайте переменные client_id, client_secret и datacenter_url и присвойте им сохраненные идентификатор клиента, секрет клиента и URL-адрес центра обработки данных из предыдущего шага:

```
 client_id = '<ваш client ID>'
>>> client_secret = '<ваш client secret>'
>>> datacenter_url = '<URL центра обработки данных Кибер Бэкап Облачный или URL адреса установки Кибер Бэкап>'
```

Выдача токена доступа клиенту API

1. Закодируйте идентификатор клиента и секретную строку клиента с помощью кодировки Base64 и сохраните результат в переменной:

```
>>> from base64 import b64encode  # Used for encoding to Base64
>>> encoded_client_creds = b64encode(f'{client_id}:{client_secret}'.encode('ascii'))
```

2. Определите переменную с именем basic_auth, а затем назначьте этой переменной объект с ключом авторизации, содержащим аутентификационные данные:

```
>>> basic_auth = {
...     'Authorization': 'Basic ' + encoded_client_creds.decode('ascii')
... }
```

3. Отправьте запрос POST на конечную точку /idp/token. Запрос должен содержать данные аутентификации в заголовках запроса и содержать поле grant_type, установленное на client_credentials в своем теле:

```
>>> response = requests.post(
...     f'{datacenter_url}/idp/token',
...     headers={'Content-Type': 'application/x-www-form-urlencoded', **basic_auth},
...     data={'grant_type': 'client_credentials'},
... )
```

4. Проверьте код состояния ответа:

```
>>> response.status_code
200
```

   Код состояния 200 означает, что на платформе произошла аутентификация клиента API и платформа выдала клиенту API токен для доступа к конечным точкам API (токен доступа). Текст тела ответа содержит закодированный объект JSON с этим токеном и некоторой другой информацией.

5. Преобразуйте текст JSON, содержащийся в теле ответа, в объект, а затем сохраните {‘access_token’: ‘eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD…’ в переменной с именем token_info:

```
>>> token_info = response.json()
>>> pprint.pprint(token_info)
'expires_on': 1562910964,
'id_token': 'eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6IjI5ZT...',
'token_type': 'bearer'}
```

6. Определите переменную с именем auth, а затем назначьте этой переменной объект, который будет использоваться для создания заголовка Authorization в запросах API:

```
>>> auth = {'Authorization': 'Bearer ' + token_info['access_token']}
>>> auth
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
```

   Вам нужно будет указывать эту переменную в каждом запросе к API следующим образом:

```
requests.get(f'{datacenter_url}/api/<api version>}/tenants', headers=auth)
```

Теперь ваша интеграция имеет доступ к соответствующему API Кибер Бэкап Облачный.

Справочник

Справочник доступен по ссылке.

2.5 Запросы и ответы API

Связь с конечными точками осуществляется через текстовые сообщения в формате HTTP, при этом само тело сообщения может быть пустым или иметь специальный формат.

Формат указывается в заголовке Content-Type сообщения.

2.5.1 Тип сообщения запроса

- Пустое тело сообщения

  Пример

```
GET /users/e5afb5e8-84b6-415b-969d-bc10d19f3301 HTTP/1.1
Host: https://installaddress.ru:433/api/v1
Authorization: Bearer dXNlcjpwYXNz
```

- Тело сообщения в формате JSON (в основном)

  Пример

```
POST /tenants HTTP/1.1
Host: https://installaddress.ru:443/api/v1
Content-Type: application/json
Authorization: Bearer dXNlcjpwYXNz
{
   "name": "The Qwerty Tenant",
   "parent_id": "fa6859a9-f5e1-4faf-a56c-5a0ae866dc4f",
   "kind": "PARTNER",
   "contact": {
      "email": "newname@example.ru",
      "address1": "Главная улица, дом 5",
      "phone": "+7 100 999 1234",
      ...
    },
   "language": "ru",
   "settings": {
   "enhanced_security": false
   }
}
```

- Тело сообщения в форме HTML

  Пример

```
POST /idp/token HTTP/1.1
Host: https://installaddress.ru:443
Content-Type: application/x-www-form-urlencoded
Authorization: Bearer dXNlcjpwYXNz

grant_type=client_credentials
```

2.5.2 Тип сообщения ответа

- Пустое тело сообщения

  Пример

  ```
  HTTP/1.1 204 No Content
  <other headers...><empty line>
  ```

- Тело сообщения в формате JSON (в основном)

  Пример

  ```
  HTTP/1.1 200 OK
  Content-Type: application/json; charset=utf-8
  Cache-Control: no-store
  Pragma: no-cache
  <other headers...>

  {
    "refresh_token":"NvLK6BAGJuNZEuQaiclVBg",
    "expires_on":1560523337,
    "token_type":"bearer",
    "access_token":"encoded.jwt.struct",
    "id_token":"identity.jwt.struct"
  }
  ```

- Тело сообщения в форме HTML

  Пример

  ```
  HTTP/1.1 200 OK
  Content-Type: text/html<other headers...>

  <!DOCTYPE html>
  <html>
  <head>
  <meta charset="UTF-8">
  </head>
  <h5>Please wait...</h5>
  </body>
  </html>
  ```

В некоторых случаях URL-адреса конечных точек могут быть дополнены параметрами запроса.

Разработчики редко создают HTTP-сообщения самостоятельно: это действие выполняет программное обеспечение, веб-браузер, прокси или веб-сервер. Хорошими инструментами для работы с такими сообщениями являются curl, PowerShell или Postman.

Например, чтобы получить информацию о тенанте из конечной точки /tenants/<tenant ID> с помощью curl, используйте следующую команду:

```
curl -isX GET https://installaddress.ru/api/v1/tenants/<your tenant ID> \
    -H "Authorization: Bearer <your token>"
```

Описания параметров curl, используемых в этом примере, приведены на странице руководства по curl.

Результат ответа может быть следующим:

```
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 17 Jun 2019 06:55:25 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 573
<other headers...>

{ "id": "f313ecf6-9256-4afd-9d47-72e032ee81d0", "version": 2, "name": "The Qwerty Tenant",
"parent_id": "fa6859a9-f5e1-4faf-a56c-5a0ae866dc4f", "kind": "PARTNER", ... }
```

Полный набор конечных точек, их описаний, поддерживаемых ими HTTP-методов, принимаемых ими данных запроса и генерируемых ими ответов описаны в справочнике API.

2.6 Параметры запросов API

Параметры запросов используются для ограничения выборки запрошенных ресурсов, поиска и фильтрации.

Операция	Описание	Рекомендуемая методика	Пример
Фильтрация (соответствие)	Отфильтровать набор ресурсов или ресурс точно соответствующие значению.	Оператор не требуется.	`?vaults=1aaahd`
Фильтрация (not)	Отфильтровать набор ресурсов или ресурс с использованием выражения "не равно" для атрибута.	Используется оператор **ne** для атрибута.	`?vaults=ne(43ddaaef)`
Фильтрация (больше или равно)	Отфильтровать набор ресурсов или ресурс с использованием выражения "больше или равно" для атрибута.	Используется оператор **ge** для атрибута.	`?enqueue_time=ge(2009-11-10T23:00:00Z)`
Фильтрация (больше)	Отфильтровать набор ресурсов или ресурс с	Используется оператор **gt** для атрибута.	`?enqueue_time=gt(2009-`

Операция	Описание	Рекомендуемая методика	Пример
	использованием выражения "больше" для атрибута.		11-10T23:00:00Z)
Фильтрация (меньше или равно)	Отфильтровать набор ресурсов или ресурс с использованием выражения "меньше или равно" для атрибута.	Используется оператор **le** для атрибута.	?enqueue_time=le(2009-11-10T23:00:00Z)
Фильтрация (меньше)	Отфильтровать набор ресурсов или ресурс с использованием выражения "меньше" для атрибута.	Используется оператор **lt** для атрибута.	?enqueue_time=lt (1568822944000000ns)
Фильтрация (внутри интервала)	Отфильтровать набор ресурсов с использованием условия "внутри интервала" для атрибута.	Используется оператор **range** для атрибута.	?enqueue_time=range (2009-11-10T23:00:00Z,2009-11-11T23:00:00Z)
Фильтрация (вне интервала)	Отфильтровать набор ресурсов с использованием условия "вне интервала" для атрибута.	Используется оператор **xrange** для атрибута.	?enqueue_time=xrange (2009-11-10T23:00:00Z,2009-11-11T23:00:00Z)
Фильтрация ("и") для значений одного атрибута	Отфильтровать набор ресурсов с использованием условия "и" для значений одного атрибута.	Используется оператор **and** для атрибута. Синтаксис: field=and (v1,v2,v3,и т.д.). Оператор **and** возможно использовать с операторами **hlike**, **like**, **tlike**.	?resources.name=and (hlike(hit), like(jar))
Фильтрация ("или") для значений одного атрибута	Отфильтровать набор ресурсов с использованием условия "или" для значений одного атрибута.	Используется оператор **or** для атрибута. Синтаксис: field=or (v1,v2,v3,и т.д.).	?resource.name=or (Critical data,CEO laptop) Будут выбраны ресурсы со значениями Critical data или CEO laptop.

Операция	Описание	Рекомендуемая методика	Пример
Фильтрация ("содержит")	Отфильтровать набор ресурсов с использованием условия "содержит" для атрибута.	Используется оператор **like** для атрибута. Синтаксис: field=like(value).	?resource.name=like(lap) Будут выбраны ресурсы со значениями laptop, thunderclap и т. д.
Фильтрация (начинается с)	Отфильтровать набор ресурсов с использованием условия "начинается с" для атрибута.	Используется оператор **hlike** для атрибута. Синтаксис: field=hlike (value).	?resource.name=hlike (CEO) Будут выбраны ресурсы со значениями CEO laptop, CEO's machines и т. д.
Фильтрация (заканчивается)	Отфильтровать набор ресурсов с использованием условия "заканчивается" для атрибута.	Используется оператор **tlike** для атрибута. Синтаксис: field=tlike (value).	?resource.name=tlike(top) Будут выбраны ресурсы со значениями laptop, tabletop и т. д.
Сортировка (по возрастанию, по убыванию)	Переопределить порядок сортировки по умолчанию набора ресурсов.	Для сортировки используется атрибут **order** с префиксами **asc** для сортировки по возрастанию и **desc** для сортировки по убыванию. Синтаксис: order=asc (f1),desc(f2).	?order=asc(date_of_ birth),desc(zip_code) Данный пример запрашивает данные с сортировкой по возрастанию по атрибуту date_of_birth в первую очередь и далее по zip_code по убыванию.

2.7 Коды ответа API

2.7.1 Проверка кодов ответов API

Ответам API присваиваются определенные числовые коды, которые позволяют быстро определить, был ли запрос к конечной точке выполнен успешно или нет, и если нет, то почему.

В отдельных главах API приведены коды ошибок и их подробные описания для каждой конечной точки API.

2.7.1.1 Проверка кодов ответов

Всем ответам API присваиваются конкретные числовые коды состояний HTTP, которые позволяют быстро определить, был ли запрос к конечной точке выполнен успешно, а если нет, то почему он

не был выполнен. Вы можете проверить коды состояний с помощью curl, добавив параметр --include:

```
curl -X POST -s <the Cyberprotect data center URL>/<Cyberprotect product API location>/status \
   --header Authorization: Bearer <your token> \
   --data "{...}"
   --include
```

Коды успеха

Код будет содержаться в первой строке вывода ответа.

Для успешного выполнения запроса платформа также может предоставить объект JSON в теле ответа, содержащий информацию об операции.

Например:

```
HTTP/1.1 200 OK
<...>
{
 "created_by": "f90e086b...",
 "last_access_at": "2020-05-22T10:13:28+00:00",
 "tenant_id": "e5afb5e8...",
 ...
}
```

Коды ошибок

Для получения кода ошибки платформа может включить в тело ответа объект JSON, содержащий информацию о конкретной ошибке. Эти коды специфичны для каждого API и конечной точки.

Значение	Описание
error.code	Код платформы или HTTP-код (см. "Коды состояния HTTP" (стр. 27)). Коды и описания платформы подробно описаны в отдельных главах, посвященных API.
error.message	Краткое сообщение с указанием причины ошибки.
error.context	Объект с дополнительной информацией об ошибке.
error.domain	Предметная область, в которой произошла ошибка. Возможные значения приведены в таблице далее.

Возможные значения error.domain в зависимости от кода ошибки:

Код ошибки	domain
400 BadRequestError	General
404 NotFoundError	
406	
409 DataConflictError	
415	
422 Issue	
500 InternalServerError	
401 UnauthorizedRequestError	Access
401 AuthenticationTimeoutError	
403 AccessDeniedError	

Например:

```
HTTP/1.1 400 Bad Request
<...>
{
  "error":
    {
      "code":1006,
      "message":"It is prohibited to delete a non-disabled tenant.",
      "context":
          {
              "id":"1415032"
          },
      "domain":"General"
   }
}
```

2.7.2 Коды состояния HTTP

Ниже приводится сводка кодов состояния HTTP, возвращаемых API, и описание каждого из них. В справочной документации по API приведен список кодов ошибок API и описаний для каждой конечной точки.

Код	Описание
20x	Запрос к конечной точке был выполнен успешно, и тело ответа может содержать объект JSON с результатами.
400	Запрос к конечной точке завершился ошибкой, и тело ответа содержит объект JSON с подробными сведениями об ошибке.

Код	Описание
401	Не удается обработать запрос к конечной точке, поскольку срок действия токена, указанного в заголовке Authorization, истек или он недействителен.
403	Не удается обработать запрос к конечной точке, поскольку ваша учетная запись не имеет права доступа к этой конечной точке.
404	Запрос к несуществующей конечной точке или в службе не найдены запрошенные данные.
405	В запросе используется метод, который не поддерживается конечной точкой.
409	Другой объект того же типа уже существует.
415	В запросе используется формат, который не поддерживается конечной точкой. Эта проблема может возникнуть, если в запросе был указан неправильный заголовок или заголовок Content-Type отсутствовал.
50x	Проблема в работе службы. Мы рекомендуем повторить запрос дважды с интервалом в 1-2 секунды. Если ошибка не устраняется, обратитесь к администратору компании или повторите попытку позже.

2.8 Сортировка и постраничные ответы

2.8.1 Постраничные ответы

Некоторые конечные точки могут выдавать большое количество результатов, что приводит к чрезмерному объему интернет-трафика и нагрузке на сервер.

Чтобы уменьшить количество результатов, эти конечные точки используют механизм постраничного разбиения, позволяющий им возвращать результаты на нескольких страницах.

Размер страницы определяется параметром limit строки запроса.

Если этот параметр явно не задан, конечная точка вернет количество результатов по умолчанию, указанное для параметра строки запроса limit в справочнике API.

Например, если имеется 99 результатов, а для параметра limit строки запроса задано значение 50 результатов, API разделит результаты на две страницы, отобразив первую страницу с 50 результатами и указателем, ведущим на вторую страницу с оставшимися 49 результатами.

Пример подобного запроса, выполняемого через curl:

```
curl -isX GET -G <the Cyberprotect data center URL>/<Cyberprotect product API
location>/<endpoint> \
    --data "limit=50" \
    -H Authorization: Bearer <your token>
```

Когда будет доступна следующая или предыдущая страница, загрузка будет включать токен страницы. Токен страницы – это строка, которая кодирует параметры запроса и позволяет перейти к следующей или предыдущей странице результатов.

Примечание
Некоторые конечные точки могут поддерживать переход только в прямом направлении, остальные – в обоих направлениях.

Когда следующая страница с результатами будет доступна, ответ будет содержать токен страницы after:

```
HTTP/1.1 200 OK
Server: nginx
Content-Type: application/json
<other headers...>

{"items":[...],"paging":{"cursors":{"after":"0A8AAAAAAAAAAA=="}}, ...}
```

Чтобы перейти к следующей странице, конечные точки могут принимать параметр строки запроса after, который должен использоваться вместе с параметром строки запроса limit. Обратите внимание, что токен страницы after должен быть закодирован в URL:

```
curl -isX GET -G <the Cyberprotect data center URL>/<Cyberprotect product API
location>/<endpoint> \
   --data "limit=50" \
   --data-urlencode "after=0A8AAAAAAAAAA==" \
   -H Authorization: Bearer <your token>
```

Когда вы перейдете на следующую страницу результатов, ответ также будет содержать токен страницы before, который позволит вам вернуться на предыдущую страницу:

```
HTTP/1.1 200 OK
Server: nginx
Content-Type: application/json
<other headers...>

{"items":[...],"paging":{"cursors":{"before":"AAAAAAAAAAAAAA=="}}, ...}
```

Для перехода на предыдущую страницу конечные точки могут принимать параметр строки запроса before, который должен использоваться вместе с параметром строки запроса limit. Обратите внимание, что токен страницы before должен быть закодирован в URL:

```
curl -isX GET -G <the Cyberprotect data center URL>/<Cyberprotect product API
location>/<endpoint> \
   --data "limit=50" \
   --data-urlencode "before=AAAAAAAAAAAAAA==" \
   -H Authorization: Bearer <your token>
```

2.8.2 Сортировка

Параметры запроса СЛЕДУЕТ использовать только с целью ограничения набора ресурсов или в качестве критериев поиска или фильтрации.

Хотя в соответствии со стандартами ограничения по URL-адресам не существует, тем не менее разные браузеры и серверы устанавливают ограничения по URL-адресам. Поэтому при большом количестве фильтруемых элементов, пожалуйста, обратите внимание на рекомендацию по использованию большого количества фильтруемых элементов.

Параметры разбивки на страницы ДОЛЖНЫ соответствовать рекомендациям по разбивке на страницы.

Порядок сортировки по умолчанию СЛЕДУЕТ рассматривать как неопределенный и недетерминированный. В этом случае сервер может выбрать для возврата результатов порядок сортировки по умолчанию.

Если требуется явный порядок сортировки, СЛЕДУЕТ использовать параметр запроса order со следующим общим синтаксисом: asc|desc(field_name),asc|desc(field_name2), где asc означает возрастание, а desc – убывание.

Пример: /accounts?order=asc(date_of_birth),desc(zip_code). В этом примере сервер запрашивает выдачу учетных записей, отсортированных сначала по параметру date_of_birth в порядке возрастания, а затем по параметру zip_code в порядке убывания.

Подробнее о фильтрации ресурсов описано на странице "Параметры запросов API" (стр. 23).

2.9 Запросы API, инициирующие длительные операции, и ожидание их окончания

Ряд сценариев выполняется в асинхронном режиме, так как их выполнение занимает длительное время. При запуске сценария в ответе может предоставляться идентификатор задачи или активности. Для получения статуса выполнения сценария необходимо выполнить запрос по идентификатору полученной задачи, активности или другой эндпоинт (см. примеры). Примеры запросов описаны ниже.

2.9.1 Примеры запросов

2.9.1.1 Запрос для запуска плана резервного копирования

Метод и путь конечной точки

```
POST /policy_applications:run
```

Заголовки

```
Accept: application/json
Content-Type: application/json
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6IjRlZDM4ZWRlLTFkZTQtNDhmNC1hM2EyLT
Q3ZGViNDdiZjEwZCJ9...
```

Тело

```
{
"state": "running",
"policy_id": "53b86e12-3c65-4721-bf31-92ddebfa769f",
"context_ids": ["5b6ff2dc-9095-4718-bc93-d25c70aaa130"]
}
```

2.9.1.2 Ответ

Код

```
202
```

Описание

```
Accepted for asynchronous run invocation
```

2.9.2 Поллинг: статус выполнения плана резервного копирования

2.9.2.1 Запрос для проверки статуса выполнения

Метод и путь конечной точки

```
GET /policy_applications?policy_id=53b86e12-3c65-4721-bf31-92ddebfa769f&context_
id=5b6ff2dc-9095-4718-bc93-d25c70aaa130
```

Заголовки

```
Accept: application/json
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6IjRlZDM4ZWRlLTFkZTQtNDhmNC1hM2EyLT
Q3ZGViNDdiZjEwZCJ9...
```

2.9.2.2 Ответ: резервное копирование в процессе

Код

```
200
OK
```

Тело запроса. Резервное копирование в процессе.


```
{
  "items": [
    [
      {
        "id": "296a5dba-d31d-4026-a21c-2cff2ecb120f",
        "created_at": "2024-11-28T09:53:05.499539125Z",
        "updated_at": "2024-11-29T06:22:51.484848419Z",
        "deleted_at": null,
        "enabled": true,
        "DEPLOYMENT": {
          "STATE": "deployed"
        },
        "RUNNING": {
          "STATE": "running",
          "PROGRESS": "20"
        },
        "last_success_time": "2024-11-28T11:13:45Z",
        "last_runtime": "2024-11-29T06:22:51Z",
        "last_activity": "policy.backup.machine",
        "next_run_time": "2024-11-29T10:01:14Z",
        "next_activity": "policy.backup.machine",
        "STATUS": "running",
        "agent_id": "bc083994-98f4-4365-97a5-7662e18ba0b2",
        "context": {
          "id": "5b6ff2dc-9095-4718-bc93-d25c70aaa130",
          "type": "resource.machine"
        },
        "policy": {
          "id": "53b86e12-3c65-4721-bf31-92ddebfa769f",
          "type": "policy.backup.machine",
          "name": "ol_test_plan_02"
        },
        "tenant_id": "17189",
        "context_tenant_id": "17190",
        "origin_contexts": [
          "5b6ff2dc-9095-4718-bc93-d25c70aaa130"
        ]
      }
    ]
  ],
  "paging": {
    "cursors": {
      "total": 1
    }
  }
}
```


2.9.2.3 Ответ: резервное копирование завершено

Код

```
200
OK
```

Тело запроса. Резервное копирование завершено.


```
{
  "items": [
    [
      {
        "id": "296a5dba-d31d-4026-a21c-2cff2ecb120f",
        "created_at": "2024-11-28T09:53:05.499539125Z",
        "updated_at": "2024-11-29T06:24:06.873787017Z",
        "deleted_at": null,
        "enabled": true,
        "DEPLOYMENT": {
          "STATE": "deployed"
        },
        "RUNNING": {
          "STATE": "idle"
        },
        "last_success_time": "2024-11-29T06:23:49Z",
        "last_runtime": "2024-11-29T06:22:38Z",
        "last_activity": "policy.backup.machine",
        "next_run_time": "2024-11-29T10:01:14Z",
        "next_activity": "policy.backup.machine",
        "STATUS": "ok",
        "agent_id": "bc083994-98f4-4365-97a5-7662e18ba0b2",
        "context": {
          "id": "5b6ff2dc-9095-4718-bc93-d25c70aaa130",
          "type": "resource.machine"
        },
        "policy": {
          "id": "53b86e12-3c65-4721-bf31-92ddebfa769f",
          "type": "policy.backup.machine",
          "name": "ol_test_plan_02"
        },
        "tenant_id": "17189",
        "context_tenant_id": "17190",
        "origin_contexts": [
          "5b6ff2dc-9095-4718-bc93-d25c70aaa130"
        ]
      }
    ]
  ],
  "paging": {
    "cursors": {
      "total": 1
    }
  }
}
```

2.10 API REST-ресурсы и операции с ними

2.10.1 Тенант

Тенант – это группа ресурсов в рамках многопользовательской системы со своими группами, пользователями и данными. Подробнее о типах тенантов на странице "Модель тенантов" (стр. 8).

Тенанты используются для разграничения данных организаций, настройки и управлением функциями системы, лицензированием.

Операции с тенантами находятся в конечной точке /tenants в API.

API представляет тенант в виде объекта JSON.

2.10.1.1 Структура объекта JSON тенанта

Описание структуры объекта JSON тенанта доступно по ссылке.

Имя	Тип значения	Описание
id	UUID string (строка UUID)	Универсальный уникальный идентификатор (UUID) тенанта. Этот UUID используется для доступа к тенанту через API.
ancestral_access	true or false (истина или ложь)	Указывает, могут ли администраторы более высокого уровня управлять тенантом.
contact	contact object (объект контактов)	Юридическая контактная информация организации.
enabled	true or false (истина или ложь)	Указывает, включен или отключен тенант.
pricing_mode	строка (string)	Режим работы тенанта. Может принимать значение TRIAL, PRODUCTION или SUSPENDED.
created_at	строка (string)	Дата и время создания тенанта по стандарту ISO 8601.
updated_at	строка (string)	Дата и время последнего обновления тенанта по стандарту ISO 8601.
deleted_at	строка (string)	Null или дата и время удаления тенанта по стандарту ISO 8601.

Имя	Тип значения	Описание
has_children	true or false (истина или ложь)	Указывает, есть ли у тенанта дочерние тенанты.
kind	string (строка)	Тип тенанта. Может принимать значение PARTNER, FOLDER, CUSTOMER или UNIT. Для персональных тенантов установите значение UNIT.
language	string (строка)	Язык уведомлений, отчетов и программного обеспечения, используемого в тенанте по умолчанию. Список поддерживаемых значений приведен ниже.
name	string (строка)	Имя тенанта.
owner_id	UUID string (строка UUID)	Установите значение UUID учетной записи пользователя, если это персональный тенант. В противном случае установите значение null.
parent_id	UUID string (строка UUID)	UUID тенанта, в котором создан этот тенант.
settings	объект (object)	Настройки тенанта.
version	number (число)	Номер версии тенанта. При каждом обновлении тенанта это число увеличивается.

2.10.1.2 Структура объекта settings

Имя	Тип значения	Описание
enhanced_security	логический (boolean)	Включает режим стандарта безопасности данных индустрии платёжных карт PCI DSS (Payment Card Industry Data Security Standard).

2.10.1.3 Структура объекта контактов в тенанте

Описание структуры объекта JSON контакта доступно по ссылке.

Имя	Тип значения	Описание
id	UUID string (строка UUID)	UUID контакта.
created_at	string	Дата и время создания контакта согласно стандарту ISO 8601.

Имя	Тип значения	Описание
	(строка)	
updated_at	string (строка)	Дата и время последнего обновления контакта согласно стандарту ISO 8601.
types	string (строка)	Возможные значения: LEGAL – контактное лицо юридического лица (компании). PRIMARY – основное контактное лицо. BILLING – контакт, который будет получать обновления о важных изменениях в отчетах об использовании платформы. У каждого тенанта может быть несколько контактов для выставления счетов. MANAGEMENT – контакт, который будет получать обновления о важных изменениях в платформе, связанных с бизнесом. У каждого тенанта может быть несколько деловых контактов. TECHNICAL – контакт, который будет получать обновления о важных технических изменениях в платформе. У каждого тенанта может быть несколько технических контактов.
title	string (строка)	Наименование организации.
website	string (строка)	Сайт организации.
industry	string (строка)	Отрасль организации.
email	string (строка)	Адрес электронной почты, который будет использоваться для активации учетной записи и служебных уведомлений.
email_ confirmed	true or false (истина или ложь)	Подтверждён ли e-mail.
address1	string (строка)	Адресная строка 1.
address2	string (строка)	Адресная строка 2.
country	string (строка)	Страна организации.
state	string (строка)	Регион организации.

Имя	Тип значения	Описание
zipcode	string (строка)	Почтовый индекс организации.
city	string (строка)	Город организации.
language	string (строка)	Язык.
phone	string (строка)	Номер телефона организации.
fax	string (строка)	Номер факса организации.
firstname	string (строка)	Имя представителя организации.
lastname	string (строка)	Фамилия представителя организации.

Поддерживаемые языковые значения

- ru – Русский;
- en, en-US – Английский.

2.10.1.4 Пример JSON объекта тенанта

```
{
   "ancestral_access": true,
   "contact": {
      "address1": "Главная улица, дом 5",
      "address2": null,
      "city": "Город",
      "country": null,
      "email": "mail@example.ru",
      "firstname": "Иван",
      "lastname": "Иванов",
      "phone": "+7 100 999 1234",
      "state": null,
      "zipcode": null,
      "title": null,
      "website": null,
      "industry": null,
      "email_confirmed": null,
      "language": null,
      "fax": null,
      ...
```

```
    },
    "enabled": true,
    "created_at": "2016-06-22T18:25:16",
    "updated_at": "2016-06-22T18:25:16",
    "deleted_at": null,
    "pricing_mode": "PRODUCTION",
    "has_children": false,
    "id": "0fcd4a69-8a40-4de8-b711-d9c83dc000f7",
    "kind": "CUSTOMER",
    "language": "ru",
    "name": "Организация",
    "owner_id": null,
    "parent_id": "ede9f834-70b3-476c-83d9-736f9f8c7dae",
    "settings": {
       "enhanced_security": false
    },
    "version": 1
}
```

2.10.1.5 Справочники

Справочники доступны по ссылке.

2.10.1.6 Действия с тенантами

Операция	Используемые методы и конечные точки
"Создание тенанта" (стр. 38)	POST /tenants
"Получение информации об отдельном тенанте" (стр. 41)	GET /tenants/{tenant_id}
"Получение списка тенантов" (стр. 42)	GET /tenants
"Получение текущего режима тенанта" (стр. 43)	GET /tenants/{tenant_id}/pricing
"Перевод тенанта в рабочий режим" (стр. 44)	PUT /tenants/{tenant_id}/pricing
"Изменение свойств тенанта" (стр. 46)	PUT /tenants/{tenant_id}
"Включение тенанта" (стр. 48)	PUT /tenants/{tenant_id}
"Отключение тенанта" (стр. 49)	PUT /tenants/{tenant_id}
"Удаление тенанта" (стр. 51)	DELETE /tenants/{tenant_id}

2.10.1.7 Создание тенанта

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
>>> tenant_id  # the UUID of the tenant to which the token provides access
'ede9f834-70b3-476c-83d9-736f9f8c7dae'
```

2. Задайте переменную tenant, а затем присвойте этой переменной объект, содержащий свойства нового тенанта:

```
>>> tenant = {
...     'name': 'Организация',
...     'kind': 'CUSTOMER',
...     'parent_id': tenant_id,
...     'language': 'ru',
...     'contact': {
...         'address1': 'Главная улица, дом 5',
...         'email': 'mail@example.ru',
...         'phone': '+7 100 999 1234',
...     },
... }
```

   Объект должен содержать ключи name, kind и parent_id.

   При указании значения для ключа kind учитывайте иерархию платформы (см. "Модель тенантов" (стр. 8)). Например, невозможно создать модульный тенант в партнерском тенанте.

   Если вы не укажете ключ language, язык тенанта по умолчанию останется английский (en).

3. Преобразуйте объект tenant в текст в формате JSON:

```
>>> tenant = json.dumps(tenant, indent=4)
```

4. Отправьте запрос POST с текстом в формате JSON на конечную точку /tenants:

```
>>> response = requests.post(
...     f'{base_url}/tenants',
...     headers={'Content-Type': 'application/json', **auth},
...     data=tenant,
... )
```

5. Проверьте код состояния запроса:

```
>>> response.status_code
201
```

   Код состояния HTTP 201 означает, что на платформе был создан тенант с указанными свойствами.

   Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, тело ответа содержит информацию о тенанте в формате JSON. При преобразовании в объект это будет выглядеть следующим образом:

```
>>> pprint.pprint(response.json())
{'ancestral_access': True,
 'contact': {
      'address1': 'Главная улица, дом 5',
      'address2': None,
      'city': None,
      'country': None,
      'email': 'mail@example.ru',
      'firstname': None,
      'lastname': None,
      'phone': '+7 100 999 1234',
      'state': None,
      'zipcode': None,
      'title': None,
      'website': None,
      'industry': None,
      'email_confirmed': None,
      'language': None,
      'fax': None,
      ... },
 'enabled': True,
 'created_at': '2016-06-22T18:25:16',
 'updated_at': "2016-06-22T18:25:16',
 'deleted_at': None,
 'pricing_mode': 'PRODUCTION',
 'has_children': False,
 'id': '0fcd4a69-8a40-4de8-b711-d9c83dc000f7',
 'kind': 'CUSTOMER',
 'language': 'ru',
 'name': 'Организация',
 'owner_id': None,
 'parent_id': 'ede9f834-70b3-476c-83d9-736f9f8c7dae',
   'settings': {
      'enhanced_security': false
   },
  'version': 1}
```

Тенанты клиентов создаются в режиме **trial**. Этот режим означает, что данные об использовании для созданного тенанта не будут включены в ежемесячные отчеты об использовании сервиса.

6. Преобразуйте текст в формате JSON, содержащийся в теле ответа, в объект и сохраните значение ключа id объекта в переменной, с названием created_tenant_id (например).

```
>>> created_tenant_id = response.json()['id']
>>> created_tenant_id
'0fcd4a69-8a40-4de8-b711-d9c83dc000f7'
```

2.10.1.8 Получение информации об отдельном тенанте

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

2. [Необязательно] Если вы хотите проверить субтенант, созданный с помощью API (см. "Создание тенанта" (стр. 38)), или субтенант, найденный по его имени, присвойте его UUID переменной tenant_id:

   ```
   >>> tenant_id = created_tenant_id
   >>> tenant_id
   '0fcd4a69-8a40-4de8-b711-d9c83dc000f7'
   ```

3. Отправьте запрос GET на конечную точку /tenants/{tenant_id}:

   ```
   >>> response = requests.get(f'{base_url}/tenants/{tenant_id}', headers=auth)
   ```

4. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   200
   ```

   Код состояния HTTP 200 означает, что текст ответа содержит информацию о тенанте в формате JSON.

   Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

5. Преобразуйте текст в формате JSON, содержащийся в теле ответа, в объект, а затем сохраните этот объект в переменной с именем tenant:

   ```
   >>> tenant = response.json()
   >>> pprint.pprint(tenant)
   {'ancestral_access': True,
    'contact': {...},
    'enabled': True,
    'created_at': '2016-06-22T18:25:16',
    'updated_at': "2016-06-22T18:25:16',
    'deleted_at': None,
    'pricing_mode': 'PRODUCTION',
    'has_children': False,
    'id': '0fcd4a69-8a40-4de8-b711-d9c83dc000f7',
   ```

```
'kind': 'CUSTOMER',
'language': 'ru',
'name': 'Организация',
'owner_id': None,
'parent_id': 'ede9f834-70b3-476c-83d9-736f9f8c7dae',
'settings': {...},
'version': 1}
```

2.10.1.9 Получение списка тенантов

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

2. Отправьте запрос GET на конечную точку /tenants:

   ```
   >>> response = requests.get(f'{base_url}/tenants', headers=auth)
   ```

3. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   200
   ```

   Код состояния HTTP 200 означает, что основной текст ответа содержит закодированный объект JSON, состоящий из элемента items. Элемент items – это массив объектов тенантов, UUID которых из списка, указанного в запросе, были найдены в платформе. Если UUID не были найдены, этот массив пуст.

   Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

4. Преобразуйте текст в формате JSON в объект, а затем сохраните значение ключа items объекта в переменной с именем tenants:

   ```
   >>> tenants = response.json()['items']
   >>> pprint.pprint(tenants)
   [{'ancestral_access': True,
    'contact': {...},
    'enabled': True,
    'has_children': False,
    'id': '0fcd4a69-8a40-4de8-b711-d9c83dc000f7',
    'kind': 'CUSTOMER',
    'language': 'ru',
   ```

```
'name': 'Организация',
'owner_id': None,
'parent_id': 'ede9f834-70b3-476c-83d9-736f9f8c7dae',
'settings': {...},
'version': 1},
{'ancestral_access': True,
'contact': {...},
'enabled': True,
'has_children': False,
'id': '14dc11ca-2b16-43bb-8ba4-2a3545c214a0',
'kind': 'CUSTOMER',
'language': 'ru',
'name': 'Организация два',
'owner_id': None,
'parent_id': 'ede9f834-70b3-476c-83d9-736f9f8c7dae',
'settings': {...},
'version': 1}]
```

2.10.1.10 Получение текущего режима тенанта

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

2. Присвойте переменной tenant_id одно из следующих значений: UUID тенанта, созданного с помощью API (см. "Создание тенанта" (стр. 38)), или тенанта, найденного по его имени:

   ```
   >>> tenant_id = created_tenant_id
   >>> tenant_id
   '0fcd4a69-8a40-4de8-b711-d9c83dc000f7'
   ```

3. Получите текущий режим тенанта, отправив запрос GET на конечную точку /tenants/{tenant_id}/pricing:

   ```
   >>> response = requests.get(f'{base_url}/tenants/{tenant_id}/pricing', headers=auth)
   ```

4. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   200
   ```

   Код состояния HTTP 200 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, текст ответа содержит объект с информацией о режиме работы тенанта в формате JSON. При преобразовании в объект он будет выглядеть следующим образом:

```
>>> pprint.pprint(response.json())
{'mode': 'PRODUCTION',
'version': 1596545222672}
```

5. Текущий режим тенанта указан в атрибуте mode.

Режим работы тенанта	Значение параметра
trial	пробный
production	рабочий
suspended	приостановленный

2.10.1.11 Перевод тенанта в рабочий режим

Тенант клиента, использующий сервисы в пробном режиме, может быть переведен в рабочий режим только один раз.

Если вы переключите режим с пробного на рабочий в середине месяца, в ежемесячный отчет об использовании сервиса будет включен весь месяц. По этой причине мы рекомендуем вам переключать режим в первый день месяца.

Режим автоматически переключается в рабочий режим, когда тенант остается в пробном режиме в течение полного месяца.

Режим тенанта доступен в настройках тенанта.

Предупреждение
Переключение режима тенанта – необратимая операция.

Пошаговая процедура

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
>>> tenant_id  # the UUID of the tenant to which the token provides access
'ede9f834-70b3-476c-83d9-736f9f8c7dae'
```

2. Присвойте переменной tenant_id одно из следующих значений: UUID тенанта, созданного с помощью API (см. "Создание тенанта" (стр. 38)), или тенанта, найденного по его имени:

```
>>> tenant_id = created_tenant_id
>>> tenant_id
'0fcd4a69-8a40-4de8-b711-d9c83dc000f7'
```

3. Получите текущий режим тенанта, отправив запрос GET на конечную точку /tenants/{tenant_id}/pricing:

```
>>> response = requests.get(f'{base_url}/tenants/{tenant_id}/pricing', headers=auth)
```

4. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что текст ответа содержит текущий режим для тенанта в формате JSON.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

5. Преобразуйте текст в формате JSON, содержащийся в теле ответа, в объект, а затем сохраните этот объект в переменной с именем tenant_pricing:

```
>>> tenant_pricing = response.json()
>>> pprint.pprint(tenant_pricing)
{
'mode': 'TRIAL',
'version': 1596545222672
}
```

6. Измените значение ключа mode на production в объекте tenant_pricing:

```
>>> tenant_pricing['mode'] = 'production'
```

7. Преобразуйте объект tenant_pricing в текст в формате JSON:

```
>>> tenant_pricing = json.dumps(tenant_pricing, indent=4)
```

8. Отправьте запрос PUT с текстом в формате JSON на конечную точку /tenants/{tenant_id}/pricing:

```
>>> response = requests.put(
...     f'{base_url}/tenants/{tenant_id}/pricing',
...     headers={'Content-Type': 'application/json', **auth},
...     data=tenant_pricing,
... )
```

9. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что произошло переключение тенанта в рабочий режим.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

2.10.1.12 Изменение свойств тенанта

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

2. Присвойте переменной tenant_id одно из следующих значений: UUID тенанта, созданного с помощью API (см. "Создание тенанта" (стр. 38)), или тенанта, найденного по его имени:

   ```
   >>> tenant_id = created_tenant_id
   >>> tenant_id
   '0fcd4a69-8a40-4de8-b711-d9c83dc000f7'
   ```

3. Получите номер последней версии тенанта. Вам необходимо будет указать этот номер в последующем запросе к тенанту. Это необходимо для управления одновременными изменениями тенанта.

   a. Извлеките информацию о тенанте, как описано в шагах 3-5 раздела "Получение информации об отдельном тенанте" (стр. 41). В результате у вас должна получиться переменная tenant с объектом тенанта.

   b. Задайте переменную tenant_version, а затем присвойте этой переменной значение version ключа из объекта тенанта:

      ```
      >>> tenant_version = tenant['version']
      >>> tenant_version
      3
      ```

4. Задайте переменную tenant_properties, а затем присвойте этой переменной объект, содержащий новые значения свойств тенанта:

   ```
   >>> tenant_properties = {
   ...     'name': 'Новая организация',
   ...     'language': 'en',
   ...     'contact': {
   ```

```
...       'address1': 'Улица новая, дом 6',
...       'email': 'newname@example.ru',
...       'phone': '+7 100 888 1234',
...    },
```

Следующие свойства тенанта могут быть безопасно изменены, не затрагивая пользователей внутри тенанта и любые данные, связанные с обслуживанием: name, kind, language, contact.

Внимание

Свойство kind может быть изменено только с partner на folder и наоборот.

5. Преобразуйте объект tenant_properties в текст в формате JSON:

```
>>> tenant_properties = {
...    'name': 'Новая организация',
...    'language': 'en',
...    'contact': {
...       'address1': 'Улица новая, дом 6',
...       'email': 'newname@example.ru',
...       'phone': '+7 100 888 1234',
...    },
```

6. Отправьте запрос PUT с текстом в формате JSON на конечную точку /tenants/{tenant_id}:

```
>>> response = requests.put(
...    f'{base_url}/tenants/{tenant_id}',
...    headers={'Content-Type': 'application/json', **auth},
...    data=tenant_properties,
... )
```

7. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что свойства указанного тенанта были обновлены на платформе.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, текст ответа содержит информацию об обновленном тенанте в формате JSON. При преобразовании в объект это будет выглядеть следующим образом:

```
>>> pprint.pprint(response.json())
{'ancestral_access': True,
 'contact': {'address1': 'Улица новая, дом 6',  # an update here
          'email': "newname@example.ru',  # an update here
          'phone': '+7 100 888 1234',  # an update here
          ...},
 'enabled': True,
```

```
'has_children': False,
'id': '0fcd4a69-8a40-4de8-b711-d9c83dc000f7',
'kind': 'CUSTOMER',
'language': 'en',  # an update here
'name': 'Новая организация',  # an update here
'owner_id': None,
'parent_id': 'ede9f834-70b3-476c-83d9-736f9f8c7dae',
...
'version': 4}  # a new revision number
```

2.10.1.13 Включение тенанта

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

2. Присвойте переменной tenant_id одно из следующих значений: UUID тенанта, созданного с помощью API (см. "Создание тенанта" (стр. 38)), или тенанта, найденного по его имени:

   ```
   >>> tenant_id = created_tenant_id
   >>> tenant_id
   '0fcd4a69-8a40-4de8-b711-d9c83dc000f7'
   ```

3. Получите номер последней версии тенанта. Вам нужно будет указать этот номер в последующем запросе к тенанту. Это необходимо для управления одновременными изменениями тенанта.
   a. Получите информацию о тенанте, как описано в шагах 3-5 раздела "Получение информации об отдельном тенанте" (стр. 41). В результате у вас должна получиться переменная tenant с объектом тенанта.
   b. Задайте переменную tenant_version, а затем присвойте этой переменной значение ключа version из объекта тенанта:

      ```
      >>> tenant_version = tenant['version']
      >>> tenant_version
      3
      ```

4. Задайте переменную tenant_update, а затем назначьте этой переменной следующий объект:

   ```
   >>> tenant_update = {
   ...     'enabled': True,
   ```

```
...     'version': tenant_version,  # this key is required
... }
```

5. Преобразуйте объект tenant_update в текст в формате JSON.

```
>>> tenant_update = json.dumps(tenant_update, indent=4)
>>> print(tenant_update)
{
   "enabled": true,
   "version": 3
}"
```

6. Отправьте запрос PUT с текстом в формате JSON на конечную точку /tenants/{tenant_id}:

```
>>> response = requests.put(
...     f'{base_url}/tenants/{tenant_id}',
...     headers={'Content-Type': 'application/json', **auth},
...     data=tenant_update,
... )
```

7. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что на платформе произошло отключение тенанта.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

2.10.1.14 Отключение тенанта

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
>>> tenant_id  # the UUID of the tenant to which the token provides access
'ede9f834-70b3-476c-83d9-736f9f8c7dae'
```

2. Присвойте переменной tenant_id одно из следующих значений: UUID тенанта, созданного с помощью API (см. "Создание тенанта" (стр. 38)), или тенанта, найденного по его имени:

```
>>> tenant_id = created_tenant_id
>>> tenant_id
'0fcd4a69-8a40-4de8-b711-d9c83dc000f7'
```

3. Получите номер последней версии тенанта. Вам нужно будет указать этот номер в последующем запросе к тенанту. Это необходимо для управления одновременными изменениями тенанта.
   a. Получите информацию о тенанте, как описано в шагах 3-5 раздела "Получение информации об отдельном тенанте" (стр. 41). В результате у вас должна получиться переменная tenant с объектом тенанта.
   b. Задайте переменную tenant_version, а затем присвойте этой переменной значение ключа version из объекта тенанта:

```
>>> tenant_version = tenant['version']
>>> tenant_version
3
```

4. Задайте переменную tenant_update, а затем назначьте этой переменной следующий объект:

```
>>> tenant_update = {
...     'enabled': False,
...     'version': tenant_version,  # this key is required
... }
```

5. Преобразуйте объект tenant_update в текст в формате JSON.

```
>>> tenant_update = json.dumps(tenant_update, indent=4)
>>> print(tenant_update)
{
   "enabled": false,
   "version": 3
}
```

6. Отправьте запрос PUT с текстом в формате JSON на конечную точку /tenants/{tenant_id}:

```
>>> response = requests.put(
...     f'{base_url}/tenants/{tenant_id}',
...     headers={'Content-Type': 'application/json', **auth},
...     data=tenant_update,
... )
```

7. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

   Код состояния HTTP 200 означает, что на платформе произошло отключение тенанта.

   Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

2.10.1.15 Удаление тенанта

Предупреждение

Удаление тенанта – необратимая операция, которая приводит к следующим результатам:

- Все субтенанты будут удалены.
- Все учетные записи пользователей, связанные с этим тенантом и всеми его субтенантами, будут удалены.
- Все службы, работающие с этим тенантом и всеми его субтенантами, перестанут работать.
- Все данные, связанные с обслуживанием (например, резервные копии, синхронизированные файлы) этого тенанта и всех его субтенантов, будут удалены.

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

2. Присвойте переменной tenant_id одно из следующих значений: UUID тенанта, созданного с помощью API (см. "Создание тенанта" (стр. 38)), или тенанта, найденного по его имени:

   ```
   >>> tenant_id = created_tenant_id
   >>> tenant_id
   '0fcd4a69-8a40-4de8-b711-d9c83dc000f7'
   ```

3. Отключите тенант, как описано в шагах 3-7 раздела "Отключение тенанта" (стр. 49).
4. Получите номер последней версии тенанта. Вам нужно будет указать этот номер в последующем запросе к тенанту. Это необходимо для управления одновременными изменениями тенанта.
   a. Получите информацию о тенанте, как описано в шагах 3-5 раздела "Получение информации об отдельном тенанте" (стр. 41). В результате у вас должна получиться переменная tenant с объектом tenant.
   b. Задайте переменную tenant_version, а затем присвойте этой переменной значение ключа version из объекта тенанта:

      ```
      >>> tenant_version = tenant['version']
      >>> tenant_version
      3
      ```

5. Отправьте запрос DELETE на конечную точку /tenants/{tenant_id}. URL-адрес конечной точки должен содержать параметр запроса version, для которого задан номер последней версии тенанта:

```
>>> params = {'version': tenant_version}
>>> response = requests.delete(f'{base_url}/tenants/{tenant_id}', headers=auth,
params=params)
```

6. Проверьте код состояния запроса:

```
>>> response.status_code
204
```

Код состояния HTTP 204 означает, что произошло удаление тенанта с платформы.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

2.10.2 Лицензия

Управление лицензированием осуществляется установкой квот на ресурсы в элементах предложения.

Элемент предложения представляет собой комбинацию некоторых логически сгруппированных функциональных возможностей, которые могут быть применены к конкретным тенантам и рабочим нагрузкам. Элементы предложения объединяют аналогичные детализированные функциональные возможности, применимые к некоторым аналогичным типам рабочих нагрузок, и являются основным лицензируемым объектом, используемым во всех ключевых сценариях лицензирования.

Квота определяет количество элементов предложения, которые возможно использовать.

Подробнее о лицензировании см. на странице "Модель лицензирования" (стр. 11).

Операции с элементами предложения и квотами находятся в конечной точке /licenses.

API представляет элемент предложения в виде объекта JSON.

2.10.2.1 Структура объекта JSON элемента предложения

Описание структуры объекта JSON элемента предложения доступно по ссылке.

Имя	Тип значения	Описание
edition	строка	Выпуск. Установлено значение null для всех элементов предложения, за исключением элементов службы Cyber Protection.
infra_id	строка UUID	UUID хранилища, относящегося к данному элементу предложения. Присутствует только в том случае, если type является infra.

Имя	Тип значения	Описание
locked	логическое значение (по умолчанию false)	Флаг, указывающий, заблокирован ли элемент предложения.
measurement_unit	строка	Единица измерения. Обратитесь к списку доступных единиц измерения (см. ниже).
name	строка	Название элемента предложения.
quota	объект quota	Объект, содержащий настройки квоты.
tenant_id	строка UUID	UUID тенанта, к которому относится данный элемент предложения.
usage_name	строка	Название метрики использования, связанного с этим элементом предложения.
updated_at	дата и время	Отметка времени последнего обновления согласно стандарту ISO 8601. Если тенант только что был создан, она равна created_at, если тенант только что был удален – deleted_at.
deleted_at	дата и время	Отметка времени мягкого удаления согласно стандарту ISO 8601.
status	строка	Возможные значения: ON, OFF.
type	строка	Тип. Ознакомьтесь со списком доступных типов элементов (см. ниже).

2.10.2.2 Структура объекта JSON квоты

Описание структуры объекта JSON квоты доступно по ссылке.

Имя	Тип значения	Описание
value	число	Значение мягкой квоты. Целое число. Может быть не задано.
overage	число	Значение жесткой квоты. Целое число. Может быть не задано.
version	число	Номер редакции объекта квоты.

2.10.2.3 Образец объекта JSON для элемента предложения


```
{
  "tenant_id": "6e6d758d-8e74-3ae3-ac84-50eb0dff12eb",
  "edition": "standard",
  "infra_id": "debe7865-fa8d-4c16-8e26-adcf8d7fd23d",
  "locked": false,
```

```
    "measurement_unit": "BYTES",
    "name": "dr_storage",
    "quota": {
        "overage": null,
        "value": null,
        "version": 0
    },
    "updated_at": "2024-11-28T11:12:43Z",
    "deleted_at": null,
    "status": "ON",
    "type": "INFRA",
    "usage_name": "dr_storage"
}
```


2.10.2.4 Доступные типы элементов

Значение	Описание
COUNT	Доступные подсчету такие элементы, как количество защищенных устройств, размер используемого облачного хранилища или количество пользователей sync & share.
FEATURE	Функции, не поддающиеся подсчету, такие как доступность локального хранилища резервных копий.
INFRA	Компоненты инфраструктуры, такие как облачное хранилище или хранилище партнеров, могут быть зарегистрированы на платформе и предлагаться в виде элемента.

2.10.2.5 Доступные единицы измерения

Значение	Описание
BYTES	Эта единица измерения используется для подсчета объема хранилища. Платформа использует двоичные байты. Это означает, что на портале управления 1073741824 байта будут отображаться как 1 ГБ.
QUANTITY	Эта единица измерения используется для подсчета виртуальных машин, мобильных устройств, рабочих станций и т. д.
N/A	Эта единица измерения используется только для типа feature, поскольку элементы предложения этого типа подсчитать нельзя.

2.10.2.6 Справочники

Справочники доступны по ссылке.

2.10.2.7 Доступные выпуски

Название	Значение	Префикс элемента	Пример
(Legacy) Cyber Backup - Standard	standard	N/A	workstations
(Legacy) Cyber Backup - Advanced	advanced	adv	adv_workstations
(Legacy) Cyber Protect - Standard	cyber_protect_std	p	p_workstations
(Legacy) Cyber Protect - Advanced	cyber_protect_adv	p_adv	p_adv_workstations
Cyber Protect (per workload)	per_workload	pw_p_ess_ (Cyber Protect Essentials functionality) pw_p_ (Cyber Protect Standard functionality) pw_p_adv_ (Cyber Protect Advanced functionality) pw_ (Cyber Backup functionality)	pw_p_ess_workstations pw_p_workstations pw_p_adv_workstations pw_workstations
Cyber Backup (per gigabyte)	per_gigabyte	pg_	pg_workstations

2.10.2.8 Действия с элементами и квотами

Операция	Используемые методы и конечные точки
"Чтение квоты" (стр. 55)	GET /licenses?tenant_id=<tenant ID>
"Задание квоты на ресурсы" (стр. 57)	POST /licenses

2.10.2.9 Чтение квоты

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
>>> tenant_id  # the UUID of the tenant to which the token provides access
'ede9f834-70b3-476c-83d9-736f9f8c7dae'
```

2. Используйте тенант, созданный через API (см. "Создание тенанта" (стр. 38)), или найдите тенант и сохраните его UUID:

```
>>> tenant_id = created_tenant_id
>>> tenant_id
'6e6d758d-8e74-3ae3-ac84-50eb0dff12eb'
```

3. Отправьте запрос GET на конечную точку /licenses?tenant_id=<tenant_id>:

```
>>> response = requests.get(
...     f'{base_url}/licenses?tenant_id={tenant_id}',
...     headers=auth,
... )
```

Примечание

По умолчанию эта конечная точка возвращает элементы предложения (устаревшей) версии Cyber Backup – Standard. Вы можете получить доступ ко всем элементам предложения, доступным для клиента или учетной записи пользователя, указав * или название редакции в параметре строки запроса edition. Для получения более подробной информации обратитесь к справочнику API.

4. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, текст ответа содержит массив items с доступными элементами предложения (см. "Лицензия" (стр. 52)) в формате JSON. При преобразовании в объект это будет выглядеть следующим образом:

```
>>> pprint.pprint(response.json())
{'items': [{
   'tenant_id': "6e6d758d-8e74-3ae3-ac84-50eb0dff12eb',
   'edition': 'standard',
   'infra_id': '019097a6-114f-4418-bd54-e01ef049f209',
   'locked': False,
   'measurement_unit': 'BYTES',
   'name': 'storage',
   'quota': {'overage': None, 'value': None, 'version': 0},
   'updated_at': "2024-10-28T11:12:43Z",
   'deleted_at': None,
   'status': 'ON',
   'type': 'INFRA',
   'usage_name': 'storage'},
  {
   'tenant_id': "6e6d758d-8e74-3ae3-ac84-50eb0dff12eb',
```

```
'edition': 'standard',
'infra_id': 'debe7865-fa8d-4c16-8e26-adcf8d7fd23d',
'locked': False,
'measurement_unit': 'BYTES',
'name': 'dr_storage',
'quota': {'overage': None, 'value': None, 'version': 0},
'updated_at': "2024-10-28T11:12:43Z",
'deleted_at': None,
'status': 'ON',
'type': 'INFRA',
'usage_name': 'dr_storage'},
...]}
```

2.10.2.10 Задание квоты на ресурсы

Квоты позволяют ограничить возможность использования сервиса. Управление квотами осуществляется с помощью **элементов предложения**, которые представляют собой набор услуг и функциональных возможностей сервиса.

Управление **квотами** для учетной записи пользователя осуществляется с помощью **персональных тенантов** учетной записи пользователя.

Существует два типа квот: "**мягкие**" и "**жесткие**".

Квота считается "**мягкой**", если вы не устанавливаете превышение квоты. Это означает, что ограничения на использование службы резервного копирования не применяются.

Квота считается “**жесткой**”, если вы устанавливаете превышение квоты. Превышение позволяет превысить квоту на указанную величину. При превышении лимита применяются ограничения на использование сервиса.

В этой процедуре в качестве примера будут использованы элементы vms, storage и workstations из расширенного выпуска.

Пошаговая процедура

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

2. Получите UUID тенанта, элементы которого и/или квоты на элементы которого вы хотите изменить.

Используйте тенант, созданный через API (см. "Создание тенанта" (стр. 38)), или найдите тенант и сохраните его UUID:

```
>>> tenant_id = created_tenant_id
>>> tenant_id
'95303d96-628c-4265-9afa-07bee3fccf39'
```

3. Получите список всех элементов advanced для тенанта, как описано в разделе "Чтение квоты" (стр. 55). В результате у вас должна быть переменная, которой присвоен список всех элементов, доступных для тенанта:


```
{'items': [{
   'tenant_id': '95303d96-628c-4265-9afa-07bee3fccf39',
   'edition': 'advanced',
   'locked': False,
   'measurement_unit': 'QUANTITY',
   'name': 'adv_workstations',
   'quota': {'overage': None, 'value': None, 'version': 0},
   'updated_at': "2024-10-28T11:12:43Z",
   'deleted_at': None,
   'status': 'ON',
   'type': 'COUNT',
   'usage_name': 'workstations'},
  {
   'tenant_id': '95303d96-628c-4265-9afa-07bee3fccf39',
   'edition': 'advanced',
   'locked': False,
   'measurement_unit': 'QUANTITY',
   'name': 'adv_vms',
   'quota': {'overage': None, 'value': None, 'version': 0},
   'updated_at': "2024-10-28T11:12:43Z",
   'deleted_at': None,
   'status': 'ON',
   'type': 'COUNT',
   'usage_name': 'vms'},
   ...]}
```


Внимание

- У отключенных элементов предложения нет поля quota.
- Ограничения квоты снимаются путем установки в полях value (мягкая квота) и overage (жесткая квота) значения None.
- Установка value (мягкая квота) равным 0 приведет к тому, что элемент будет недоступен тенанту, поскольку квота уже будет превышена.
- Установка ограничения value (мягкой квоты) на None приведет к сбросу version на 0, а overage – на None.
- Значение version должно быть тем же самым, что и в ранее выбранном объекте элемента.

4. Просмотрите список элементов и установите мягкую квоту для adv_vms на 15 виртуальных машин, а для adv_workstations – на 10 рабочих станций.

```
>>> for offering_item in offering_items:
...     # Check if the offering item is enabled
...     if offering_item['status']:
...         if offering_item['name'] == 'adv_workstations':
...             offering_item['quota']['value'] = 10  #  measurement_unit - quantity
...         elif offering_item['name'] == 'adv_vms':
...             offering_item['quota']['value'] = 15  #  measurement_unit - quantity
```

5. Задайте переменную updated_offering_items и присвойте ей объект с ключом offering_items со списком элементов предложения:

```
>>> updated_offering_items = {
...     'offering_items': offering_items
... }
```

6. Конвертируйте объект updated_offering_items в текст JSON:

```
>>> updated_offering_items = json.dumps(updated_offering_items, indent=4)
```

7. Отправьте запрос POST с текстом JSON на конечную точку /licenses:

```
>>> response = requests.post(
...     f'{base_url}/licenses',
...     headers={'Content-Type': 'application/json', **auth},
...     data=updated_offering_items,
... )
```

8. Проверьте код ответа:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что квоты элементов предложений были изменены.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, тело ответа будет содержать массив с обновленным элементом предложения в формате JSON. После конвертации в объект он будет выглядеть так:

```
>>> pprint.pprint(response.json())
{'items': [{
   'tenant_id': '95303d96-628c-4265-9afa-07bee3fccf39',
   'edition': 'advanced',
   'locked': False,
   'measurement_unit': 'QUANTITY',
   'name': 'adv_workstations',
   'quota': {'overage': None, 'value': 10, 'version': 1565794018233},
   'updated_at': "2024-10-28T11:12:43Z",
```

```
 'deleted_at': None,
 'status': 'ON',
 'type': 'COUNT',
 'usage_name': 'workstations'},
{
 'tenant_id': '95303d96-628c-4265-9afa-07bee3fccf39',
 'edition': 'advanced',
 'locked': False,
 'measurement_unit': 'QUANTITY',
 'name': 'adv_vms',
 'quota': {'overage': None, 'value': 15, 'version': 1565794018233},
 'updated_at': "2024-10-28T11:12:43Z",
 'deleted_at': None,
 'status': 'ON',
 'type': 'COUNT',
 'usage_name': 'vms'},
 ...]}
```

2.10.3 Использование

Использование (**потребление**) – информация об использовании (потреблении) услуг тенантом: объём хранения для облачного хранилища или количество защищённых рабочих нагрузок (виртуальные машины, физические машины, приложения).

Операции с использованиями находятся в конечной точке /usages.

2.10.3.1 Структура объекта JSON использования

Описание структуры объекта JSON использования доступно по ссылке.

Имя	Тип значения	Описание
tenant_id	строка UUID	UUID тенанта, к которому относится данное использование (см. "Тенант" (стр. 34)).
edition	строка	Версия использования.
infra_id	строка UUID	UUID хранилища, относящегося к данному элементу. Присутствует только в том случае, если type указан как infra.
measurement_unit	строка	Единица измерения (см. "Лицензия" (стр. 52)).
name	строка	Название для использования с префиксом издания.
offering_item	объект	Объект, содержащий квоту и статус элемента предложения. Присутствует только в том случае, если использование связано с элементом (см. "Лицензия" (стр. 52)).
range_start	строка	Дата и время по стандарту ISO 8601 с момента начала сбора данных об использовании. По умолчанию это первый день

Имя	Тип значения	Описание
		текущего месяца.
usage_name	строка	Название для использования без префикса редакции.
value	число	Текущее значение использования.
absolute_value	число	Накопленное значение использования.
type	строка	Тип использования. Поскольку эти типы используются совместно с элементами предложения, ознакомьтесь со списком доступных типов элементов (см. "Лицензия" (стр. 52)).

2.10.3.2 Образец объекта JSON для использования


```
{
  "absolute_value": 0,
  "edition": "standard",
  "infra_id": "019097a6-114f-4418-bd54-e01ef049f209",
  "measurement_unit": "BYTES",
  "name": "storage",
  "usage_name": "storage",
  "offering_item": {
    "quota": {
      "overage": null,
      "value": null,
      "version": 0
    },
    "status": "ON"
  },
  "range_start": "2024-08-01T00:00:00",
  "tenant_id": "95303d96-628c-4265-9afa-07bee3fccf39",
  "type": "infra",
  "value": 0
}
```


2.10.3.3 Справочники

Справочники доступны по ссылке.

2.10.3.4 Действия с использованиями

Операция	Используемые методы и конечные точки
"Получение объема хранения для облачного хранилища" (стр. 62)	GET /usages?tenant_id=<tenant ID>
"Получение количества защищенных рабочих нагрузок " (стр. 63)	GET /usages?tenant_id=<tenant ID>

2.10.3.5 Получение объема хранения для облачного хранилища

Тенанты и **персональные тенанты** предоставляют показатели использования сервиса на уровне элементов предложения.

Персональный тенант представляет собой тенант (см. "Тенант" (стр. 34)), привязанный к определенной учетной записи пользователя (см. "Пользователь" (стр. 65)), и используется только для контроля квот учетных записей пользователей и сбора данных о реальном использовании сервиса учетной записью пользователя.

Пошаговая процедура

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

2. Определите UUID тенанта, для которого требуется информация по использованию, с помощью поиска (см. "Поиск пользователя" (стр. 74)):

   ```
   >>> tenant_id = user_tenant_id
   >>> tenant_id
   'e18a5b6f-5ba4-44b0-8b41-033148877aee'
   ```

3. Для получения объёма хранения облачного хранилища требуется передавать параметр usage_names.

   ```
   >>>usage_names='storage'
   ```

   Отправьте запрос GET на конечную точку /usages:

   ```
   >>> response = requests.get(f'{base_url}/usages?tenant_id={tenant_id}&usage_names=
   {usage_names}', headers=auth)
   ```

4. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   200
   ```

   Код состояния HTTP 200 означает, что запрос был успешно выполнен.

   Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, текст ответа содержит информацию об использовании тенанта в формате JSON. При преобразовании в объект он будет выглядеть следующим образом:

```
>>> pprint.pprint(response.json())
{   "items": [
                'name': 'storage',
                'edition': 'standard',
                'usage_name': 'storage',
                'type': 'INFRA',
                'measurement_unit': 'BYTES',
                'infra_id': '019097a6-114f-4418-bd54-e01ef049f209',
                'tenant_id': 'e18a5b6f-5ba4-44b0-8b41-033148877aee',
                'range_start': '2019-08-01T00:00:00',
                'absolute_value': 139874690,
                'value': 139874690,
                'offering_item': {
                    'quota': {'overage': None,
                              'value': None,
                              'version': 0},
                    'status': 'ON'
    ]
}
```

2.10.3.6 Получение количества защищенных рабочих нагрузок

Тенанты и **персональные тенанты** предоставляют показатели использования сервиса на уровне элементов предложения.

Персональный тенант представляет собой тенант (см. "Тенант" (стр. 34)), привязанный к определенной учетной записи пользователя (см. "Пользователь" (стр. 65)), и используется только для контроля квот учетных записей пользователей и сбора данных о реальном использовании сервиса учетной записью пользователя.

Пошаговая процедура

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

2. Определите UUID тенанта, для которого требуется информация по использованию, с помощью поиска (см. "Поиск пользователя" (стр. 74)):

```
>>> tenant_id = user_tenant_id
>>> tenant_id
'e18a5b6f-5ba4-44b0-8b41-033148877aee'
```

3. Для получения количества защищённых устройств требуется передавать параметр usage_ names.

Тип устройства	Значение параметра
Рабочая станция	workstations
Сервер	servers
Виртуальная машина	vms
Сервер веб-хостинга	web_hosting_servers
Экземпляр СУБД PostgreSQL	postgresql
Почтовые ящики	mailboxes

```
>>>usage_names='postgresql'
```

Отправьте запрос GET на конечную точку /usages:

```
>>> response = requests.get(f'{base_url}/usages?tenant_id={tenant_id}&usage_names=
{usage_names}', headers=auth)
```

4. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, текст ответа содержит информацию об использовании тенанта в формате JSON. При преобразовании в объект он будет выглядеть следующим образом:

```
>>> pprint.pprint(response.json())
{   "items": [
                {
                'name': 'p_adv_postgresql',
                'edition': 'cyber_protect_adv',
                'usage_name': 'postgresql',
                'type': 'COUNT',
                'measurement_unit': 'QUANTITY',
                'tenant_id': 'e18a5b6f-5ba4-44b0-8b41-033148877aee',
                'range_start': '2019-08-01T00:00:00',
                'absolute_value': 40,
```

```
                'value': 40,
                'offering_item': {
                    'quota': {'overage': None,
                              'value': None,
                              'version': 0},
                    'status': 'ON'}
                }
            ]
}
```

2.10.4 Пользователь

Учетная запись пользователя – это учетная запись человека или системы, которая имеет доступ к облачному сервису.

Она используется для следующего:

- разграничение доступа к функциям и данным системы;
- сбор данных об объёме использования ресурсов;
- получение оповещений и отчётов;
- интеграция со сторонними системами.

Все операции с **учетными записями пользователей** в облачной платформе находятся в конечной точке /users.

Учетные записи пользователей, созданные в клиентском тенанте, имеют **персональный тенант**. **Персональный тенант** представляет собой тенант, привязанный к определенной учётной записи пользователя, и используется **только** для контроля квот учётной записи пользователя и сбора данных о реальном использовании учётной записи пользователя.

2.10.4.1 Структура объекта JSON учётной записи пользователя

Описание структуры объекта JSON учётной записи пользователя доступно по ссылке.

Имя	Тип значения	Описание
id	строка UUID	UUID учётной записи пользователя.
version	число	Номер редакции.
tenant_id	строка UUID	UUID тенанта (см. "Тенант" (стр. 34)), в котором была создана учетная запись пользователя.
login	строка	Логин учётной записи пользователя.
contact	объект контактов	Контактная информация учётной записи.
activated	логическое	Статус активации пользователя.

Имя	Тип значения	Описание
	значение (по умолчанию: ложь)	
enabled	логическое значение (по умолчанию: истина)	Флаг, который отключает или включает учетную запись пользователя на платформе.
created_at	строка	Дата и время создания пользователя согласно стандарту ISO 8601.
language	строка	Язык пользователя.
updated_at	строка	Дата и время последнего обновления пользователя согласно стандарту ISO 8601.
deleted_at	строка	Дата и время удаления пользователя согласно стандарту ISO 8601 или null в случае, если пользователь не удалён.
personal_ tenant_id	строка	UUID персонального тенанта (если пользователь был создан в клиентском тенанте). Этот тенант включает контактную информацию учётной записи пользователя и имеет значение kind, равное unit.
business_ types	строка	Возможные значения: BUYER.

2.10.4.2 Структура объекта контактов учётной записи пользователя

Описание структуры объекта JSON контактов учётной записи пользователя доступно по ссылке.

Имя	Тип значения	Описание
id	UUID string (строка UUID)	UUID контакта.
created_at	string (строка)	Дата и время создания контакта согласно стандарту ISO 8601.
updated_at	string (строка)	Дата и время последнего обновления контакта согласно стандарту ISO 8601.
types	string (строка)	Возможные значения: LEGAL – контактное лицо юридического лица (компании). PRIMARY – основное контактное лицо. BILLING – контакт, который будет получать обновления о важных изменениях в отчетах об использовании платформы. У каждого тенанта

Имя	Тип значения	Описание
		может быть несколько контактов для выставления счетов. MANAGEMENT – контакт, который будет получать обновления о важных изменениях в платформе, связанных с бизнесом. У каждого тенанта может быть несколько деловых контактов. TECHNICAL – контакт, который будет получать обновления о важных технических изменениях в платформе. У каждого тенанта может быть несколько технических контактов.
title	string (строка)	Наименование организации.
website	string (строка)	Сайт организации.
industry	string (строка)	Отрасль организации.
email	string (строка)	Адрес электронной почты, который будет использоваться для активации учётной записи и служебных уведомлений.
email_ confirmed	true or false (истина или ложь)	Подтверждён ли e-mail.
address1	string (строка)	Адресная строка 1.
address2	string (строка)	Адресная строка 2.
country	string (строка)	Страна организации.
state	string (строка)	Регион организации.
zipcode	string (строка)	Почтовый индекс организации.
city	string (строка)	Город организации.
language	string (строка)	Язык.
phone	string (строка)	Номер телефона организации.

Имя	Тип значения	Описание
fax	string (строка)	Номер факса организации.
firstname	string (строка)	Имя представителя организации.
lastname	string (строка)	Фамилия представителя организации.

2.10.4.3 Пример учётной записи пользователя


```
{
  "id": "948efcf2-b740-4c40-bb2d-4e4a46adfd87",
  "version": 2,
  "tenant_id": "0ef03214-6e47-4e50-87f2-a5955ba6095c",
  "login": "mylogin",
  "contact": {
    "id": "6e6d758d-8e74-3ae3-ac84-50eb0dff12eb",
    "email": "newname@example.ru",
    "address1": "Главная улица, дом 5",
    "address2": "",
    "country": "",
    "state": "",
    "zipcode": "",
    "city": "",
    "phone": "+7 100 999 1234",
    "firstname": "",
    "lastname": "",
    "types": ["BILLING", "MANAGEMENT", "TECHNICAL"],
    "title": "",
    "website": "",
    "industry": "",
     "email_confirmed": false,
    "language": "",
    "fax": ""  },
  "activated": true,
  "enabled": true,
  "created_at": "2019-07-25T07:11:02.807354+00:00",
  "updated_at": "2019-07-25T07:11:02.807354+00:00",
  "deleted_at": "",
  "language": "ru",
  "personal_tenant_id": "2f8ad2e2-28f2-11e7-aad1-5ffe2ad47151",
  "business_types": []
}
```


2.10.4.4 Справочники

Справочники доступны по ссылке.

2.10.4.5 Действия с учетными записями пользователей

Операция	Используемые методы и конечные точки
"Проверка доступности имени пользователя" (стр. 69)	GET /users:check_login?username=<user login>
"Поиск пользователя" (стр. 74)	GET /users:search
Создание пользователя для тенанта (см. "Создание учетной записи пользователя" (стр. 70))	POST /users
"Отправка активационной ссылки пользователю" (стр. 73)	POST /users/<user ID>:send_activation_email
"Получение списка пользователей тенанта" (стр. 78)	GET /users
"Получение информации о пользователе" (стр. 76)	GET /users/<user ID>
"Изменение свойств пользователя" (стр. 80)	PUT /users/<user ID>
"Изменение электронной почты учетной записи пользователя" (стр. 80)	PUT /users/{user_id}
"Изменение контактной информации учетной записи пользователя" (стр. 81)	PUT /users/{user_id}
"Удаление пользователя" (стр. 83)	DELETE /users/<user ID>

2.10.4.6 Проверка доступности имени пользователя

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

2. Определите переменную с параметром user_login:

   ```
   >>> user_login = "JohnDoe"  # username as login
   ```

Имя	Тип значения	Необходимо	Описание
username	строка	Да	Имя пользователя для проверки доступности.

3. Проверьте, доступно ли имя учетной записи, отправив запрос GET на конечную точку /users:check_login?username=<user_login>:

```
>>> response = requests.get(f'{base_url}/users:check_login?username={user_login}',
headers=auth)
```

4. Проверьте код состояния запроса:

```
>>> response.status_code
404
```

Код состояния	Результат
204	Указанное имя используется учетной записью на платформе.
404	Указанное имя не используется никакими учетными записями на платформе.

Код состояния HTTP 204 означает, что указанное имя не используется никакими другими учетными записями на платформе.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

2.10.4.7 Создание учетной записи пользователя

Внимание

Платформа применяет следующие правила для входа в учетную запись пользователя:

- Длина имени учетной записи пользователя (логина) должна составлять не менее 3 символов. В противном случае найти его с помощью /search будет невозможно.
- Имя учетной записи пользователя может содержать буквы ASCII, цифры и следующие специальные символы: .@_-+:!#$%^*={}"/?.
- Имя учетной записи пользователя не должно содержать пробелов.
- При входе в учетную запись пользователя в качестве имени может использоваться адрес электронной почты.

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
>>> tenant_id  # the UUID of the tenant to which the token provides access
'ede9f834-70b3-476c-83d9-736f9f8c7dae'
```

2. Проверьте доступность имени пользователя для регистрации (см. "Проверка доступности имени пользователя" (стр. 69)).
3. Задайте переменную user_data, а затем присвойте этой переменной информацию об учетной записи пользователя (см. "Пользователь" (стр. 65)):

```
>>> user_data = {
...     "tenant_id": tenant_id,
...     "login": login,
...     "contact": {
...         "email": "johndoe@mysite.com",
...         "firstname": "John",
...         "lastname": "Doe",
...         "types": ["billing"]
...     }
... }
```

Имя	Тип значения	Необходимо	Описание
tenant_id	строка UUID	Да	UUID тенанта, для которого будет создана учетная запись пользователя.
login	строка	Да	Имя учетной записи пользователя.
contact	объект контактов (см. "Пользователь" (стр. 65))	Да	Контактная информация учетной записи. Требуется параметр email.

4. Преобразуйте объект user_data в текст в формате JSON.

```
>>> user_data = json.dumps(user_data, indent=4)
```

5. Отправьте запрос POST с текстом в формате JSON на конечную точку /users:

```
>>> response = requests.post(
...     f'{base_url}/users',
...     headers={'Content-Type': 'application/json', **auth},
...     data=user_data,
... )"
```

6. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что платформа создала неактивированную учетную запись пользователя с именем JohnDoe. Если пользователь был создан в клиентском тенанте, для этого пользователя также будет создан персональный тенант.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, текст ответа содержит информацию об учетной записи пользователя (см. "Пользователь" (стр. 65)) в формате JSON. При преобразовании в объект он будет выглядеть следующим образом:

```
>>> pprint.pprint(response.json())
{'activated': False,
 'business_types': [],
 'contact': {
        'address1': '',
        'address2': '',
        'city': '',
        'country': '',
        'email': 'johndoe@mysite.com',
        'firstname': 'John',
        'lastname': 'Doe',
        'phone': '',
        'state': '',
        'zipcode': '',
        'id': '6e6d758d-8e74-3ae3-ac84-50eb0dff12eb',
        'types': [],
        'title': '',
        'website': '',
        'industry': '',
        'email_confirmed": false,
        'language': '',
        'fax': '',
        'created_at': '2019-07-25T07:11:02.807354+00:00',
        'updated_at': '2019-07-25T07:11:02.807354+00:00'
},
 'created_at': '2019-07-25T07:11:02.807354+00:00',
 'updated_at': '2019-07-25T07:11:02.807354+00:00',
 'deleted_at': None,
 'enabled': True,
 'id': '1c234e69-5469-424a-a6d1-ff5658b387a6',
 'language': 'en',
 'login': 'JohnDoe',
 'personal_tenant_id': None,
 'tenant_id': 'ede9f834-70b3-476c-83d9-736f9f8c7dae',
 'version': 1}
```

7. Сохраните номер редакции и UUID учетной записи пользователя, которые потребуются для управления учетной записью пользователя:

```
>>> user_id = response.json()['id']
>>> user_id
'1c234e69-5469-424a-a6d1-ff5658b387a6'
>>> version = response.json()['version']
>>> version
1
```

Если эта учетная запись пользователя создана в клиентском тенанте, то также сохраните UUID ее персонального тенанта в переменной:

```
>>> personal_tenant_id = response.json()['personal_tenant_id']
>>> personal_tenant_id
'e18a5b6f-5ba4-44b0-8b41-033148877aee'
```

Далее необходимо активировать новую учетную запись пользователя (см. "Отправка активационной ссылки пользователю" (стр. 73)).

2.10.4.8 Отправка активационной ссылки пользователю

Внимание
Платформа применяет следующие правила для конечной точки /users/<user_id>:send_activation_email:

- Только клиенты API могут получить доступ к этой конечной точке.
- Электронное письмо с активацией может быть отправлено только учетным записям пользователей, расположенным в субтенантах.

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

2. Присвойте переменной user_id UUID учетной записи пользователя, созданной с помощью API (см. "Создание учетной записи пользователя" (стр. 70)), или учетной записи пользователя, найденной с помощью поиска (см. "Поиск пользователя" (стр. 74)):

   ```
   >>> user_id = created_user_id
   >>> user_id
   '1c234e69-5469-424a-a6d1-ff5658b387a6'
   ```

3. Отправьте запрос POST на конечную точку /users/<user_id>:send_activation_email:

```
>>> response = requests.post(f'{base_url}/users/{user_id}:send_activation_email', headers=
{'Content-Type': 'application/json', **auth})
```

Для /users/<user_id>:send_activation_email требуется пустой текст в формате JSON.

4. Проверьте код состояния запроса:

```
>>> response.status_code
204
```

Код состояния HTTP 204 означает, что письмо со ссылкой для активации отправлено.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

2.10.4.9 Поиск пользователя

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
>>> tenant_id  # the UUID of the tenant to which the token provides access
'ede9f834-70b3-476c-83d9-736f9f8c7dae'
```

2. Отправьте запрос GET на конечную точку /users:search. URL-адрес конечной точки должен содержать параметр запроса tenant_id, равный UUID клиента, с которого будет начинаться поиск пользователей, и параметр запроса text, по которому будет выполняться поиск среди атрибутов учётных записей:

```
>>> account_login = 'foobar'
>>> params = {'tenant_id': tenant_id, 'text': account_login}
>>> response = requests.get(f'{base_url}/users:search', headers=auth, params=params)
```

3. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что основной текст ответа содержит закодированный объект JSON, состоящий из элемента items. Элемент items представляет собой массив результирующих объектов. Если ресурсы, содержащие указанный текст, не найдены, этот массив пуст.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

4. Преобразуйте текст в формате JSON в объект, а затем сохраните значение ключа items объекта в переменной с именем results:

```
>>> results = response.json()['items']
>>> pprint.pprint(results)
[{'first_name': '',
  'id': '0fcd4a69-8a40-4de8-b711-d9c83dc000f7',
  'kind': 'CUSTOMER',
  'last_name': '',
  'name': 'Foobar',  # a match here
  'obj_type': 'TENANT'
  'parent_id': 'ede9f834-70b3-476c-83d9-736f9f8c7dae',
  'path': ['JohnSmith, Inc', 'Foobar']},
 {'first_name': 'John',
  'id': '2ae8a1e9-4dba-4a07-b711-d9c83dc000f7',
  'last_name': 'Doe',
  'login': 'JohnDoe',
  'obj_type': 'USER'
  'parent_id': '0fcd4a69-8a40-4de8-b711-d9c83dc000f7',
  'path': ['JohnSmith, Inc', 'Foobar']},  # a match in the email address
 {'first_name': '',
  'id': '2ae8a1e9-4dba-4a07-b56a-c51cec1485fc',
  'last_name': '',
  'login': 'foobar',  # a match here
  'obj_type': 'USER',
  'parent_id': '0fcd4a69-8a40-4de8-b711-d9c83dc000f7',
  'path': ['JohnSmith, Inc', 'Foobar']}]
```

5. Найдите объект, в котором значение ключа obj_type равно user, а значение ключа login равно имени искомой учетной записи пользователя (account_login):

```
>>> found_account = None
>>> for res in results:
...     if res['obj_type'] == 'user' and res['login'] == account_login:
...         found_account = res
...         break
... else:
...     print('A user account with this login is not found.')
... .
>>> pprint.pprint(found_account)
{'first_name': '',
 'id': '2ae8a1e9-4dba-4a07-b56a-c51cec1485fc',
 'last_name': '',
 'login': 'foobar',
 'obj_type': 'USER',
 'parent_id': '0fcd4a69-8a40-4de8-b711-d9c83dc000f7',
 'path': ['JohnSmith, Inc', 'Foobar']}
```

Имя	Тип значения	Описание
obj_type	строка	Тип ресурса, который представляет этот объект.
id	строка	UUID учетной записи.
login	строка	Имя учетной записи.
parent_id	строка	UUID тенанта, где зарегистрирована эта учетная запись.
first_ name	строка	Значение firstname из объекта contact учетной записи.
last_ name	строка	Значение lastname из объекта contact учетной записи.
path	массив строк	Путь к тенанту, UUID которого указан в элементе parent_id, относительно тенанта, с которого начался поиск.

2.10.4.10 Получение информации о пользователе

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

2. Присвойте переменной user_id UUID учетной записи пользователя, созданной с помощью API (см. "Создание учетной записи пользователя" (стр. 70)), или учетной записи пользователя, найденной с помощью поиска (см. "Поиск пользователя" (стр. 74)):

   ```
   >>> user_id = created_user_id
   >>> user_id
   '1c234e69-5469-424a-a6d1-ff5658b387a6'
   ```

3. Отправьте запрос GET на конечную точку /users/<user_id>:

   ```
   >>> response = requests.get(f'{base_url}/users/{user_id}', headers=auth)
   ```

4. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   200
   ```

   Код состояния HTTP 200 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, тело ответа содержит информацию об учетной записи пользователя в формате JSON. При преобразовании в объект это будет выглядеть следующим образом:

```
>>> pprint.pprint(response.json())
{'activated': False,
 'business_types': [],
 'contact': {
          'address1': '',
          'address2': '',
          'city': '',
          'country': '',
          'email': 'johndoe@mysite.com',
          'firstname': 'John',
          'lastname': 'Doe',
          'phone': '',
          'state': '',
          'zipcode': '',
          'id': '6e6d758d-8e74-3ae3-ac84-50eb0dff12eb',
          'types': [],
          'title': '',
          'website': '',
          'industry': '',
          'email_confirmed'': false,
          'language': '',
          'fax': '',
          'created_at': '2019-07-25T07:11:02.807354+00:00',
          'updated_at': '2019-07-25T07:11:02.807354+00:00'
},
 'created_at': '2019-07-25T07:11:02.807354+00:00',
 'updated_at': '2019-07-25T07:11:02.807354+00:00',
 'deleted_at': None,
 'enabled': True,
 'id': '1c234e69-5469-424a-a6d1-ff5658b387a6',
 'language': 'en',
 'login': 'JohnDoe',
 'personal_tenant_id': None,
 'tenant_id': 'ede9f834-70b3-476c-83d9-736f9f8c7dae',
 'version': 1}
```

5. [Необязательно] Сохраните номер редакции учетной записи пользователя на случай, если вам потребуется изменить или удалить ее:

```
>>> version = response.json()['version']
>>> version
1
```

2.10.4.11 Получение списка пользователей тенанта

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

2. Отправьте запрос GET на конечную точку /users:

   ```
   >>> response = requests.get(f'{base_url}/users', headers=auth)
   ```

3. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   200
   ```

   Код состояния HTTP 200 означает, что основной текст ответа содержит закодированный объект JSON, состоящий из элемента items. Элемент items представляет собой массив результирующих объектов. Если ресурсы, содержащие указанный текст, не найдены, этот массив пуст.

   Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

   Преобразуйте текст в формате JSON в объект, а затем сохраните значение ключа items объекта в переменной с именем results:

   ```
   >>> results = response.json()['items']
   >>> pprint.pprint(results)
   [{'activated': False,
    'business_types': [],
    'contact': {
              'address1': '',
              'address2': '',
              'city': '',
              'country': '',
              'email': 'johndoe@mysite.com',
              'firstname': 'John',
              'lastname': 'Doe',
              'phone': '',
              'state': '',
              'zipcode': '',
              'id': '6e6d758d-8e74-3ae3-ac84-50eb0dff12eb',
              'types': [],
              'title': '',
   ```

```
            'website': '',
            'industry': '',
            'email_confirmed": false,
            'language': '',
            'fax': '',
            'created_at': '2019-07-25T07:11:02.807354+00:00',
            'updated_at': '2019-07-25T07:11:02.807354+00:00'
},
 'created_at': '2019-07-25T07:11:02.807354+00:00',
 'updated_at': '2019-07-25T07:11:02.807354+00:00',
 'deleted_at': None,
 'enabled': True,
 'id': '1c234e69-5469-424a-a6d1-ff5658b387a6',
 'language': 'en',
 'login': 'JohnDoe',
 'personal_tenant_id': None,
 'tenant_id': 'ede9f834-70b3-476c-83d9-736f9f8c7dae',
 'version': 1},
{'activated': True,
 'business_types': [],
 'contact': {
            'address1': '',
            'address2': '',
            'city': '',
            'country': '',
            'email': 'johndoe2@mysite.com',
            'firstname': 'John',
            'lastname': 'Doe',
            'phone': '',
            'state': '',
            'zipcode': '',
            'id': 'cc29685d-a9ca-3e87-a83d-069f18b588f0',
            'types': [],
            'title': '',
            'website': '',
            'industry': '',
            'email_confirmed": false,
            'language': '',
            'fax': '',
            'created_at': '2019-07-25T07:11:02.807354+00:00',
            'updated_at': '2019-07-25T07:11:02.807354+00:00'
},
 'created_at': '2019-07-25T07:11:02.807354+00:00',
 'updated_at': '2019-07-25T07:11:02.807354+00:00',
 'deleted_at': None,
 'enabled': True,
 'id': 'a66a5983-3c0c-444e-8f7b-c08c5438f562',
 'language': 'en',
 'login': 'JohnDoe',
 'personal_tenant_id': None,
 'tenant_id': 'ede9f834-70b3-476c-83d9-736f9f8c7dae',
 'version': 1}]
```

2.10.4.12 Изменение свойств пользователя

Изменение электронной почты учетной записи пользователя

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

2. Присвойте переменной user_id UUID учетной записи пользователя, созданной с помощью API (см. "Создание учетной записи пользователя" (стр. 70)), или учетной записи пользователя, найденной с помощью поиска (см. "Поиск пользователя" (стр. 74)):

   ```
   >>> user_id = created_user_id
   >>> user_id
   '1c234e69-5469-424a-a6d1-ff5658b387a6'
   ```

3. Получите номер редакции учетной записи пользователя, как описано в предыдущих главах. Следующая переменная должна быть доступна уже сейчас:

   ```
   >>> version
   1
   ```

4. Задайте переменную user_data, а затем назначьте этой переменной адрес электронной почты учетной записи пользователя для обновления (см. "Пользователь" (стр. 65)):

   ```
   >>> user_data = {
   ...     "contact": {
   ...         "email": "johndoe@newmail.net"
   ...     },
   ...     "version": version
   ... }
   ```

Имя	Тип значения	Обязательно	Описание
contact	объект контактов (см. "Пользователь" (стр. 65))	Нет	Контактная информация учетной записи.
version	число	Да	Номер редакции.

5. Преобразуйте объект user_data в текст в формате JSON:

```
>>> user_data = json.dumps(user_data, indent=4)
```

6. Отправьте запрос PUT с текстом в формате JSON на конечную точку /users/<user_id>:

```
>>> response = requests.put(
...     f'{base_url}/users/{user_id}',
...     headers={'Content-Type': 'application/json', **auth},
...     data=user_data,
... )
```

7. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что адрес электронной почты был изменен, если учетная запись пользователя **не была активирована**, или ссылка для подтверждения по электронной почте была отправлена на старый адрес электронной почты учетной записи пользователя, если учетная запись пользователя **была активирована**.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Внимание

- Адрес электронной почты будет изменен только после того, как пользователь откроет ссылку для подтверждения отправки по электронной почте.
- Номер редакции будет увеличен, только если адрес электронной почты был изменен для **неактивированной** учетной записи пользователя.

Изменение контактной информации учетной записи пользователя

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
>>> tenant_id  # the UUID of the tenant to which the token provides access
'ede9f834-70b3-476c-83d9-736f9f8c7dae'
```

2. Присвойте переменной user_id UUID учетной записи пользователя, созданной с помощью API (см. "Создание учетной записи пользователя" (стр. 70)), или учетной записи пользователя, найденной с помощью поиска (см. "Поиск пользователя" (стр. 74)):

```
>>> user_id = created_user_id
>>> user_id
'1c234e69-5469-424a-a6d1-ff5658b387a6'
```

3. Получите номер редакции учетной записи пользователя, как описано в предыдущих главах. Следующая переменная должна быть доступна уже сейчас:

```
>>> version
1
```

4. Задайте переменную user_data, а затем назначьте контактную информацию учетной записи пользователя (см. "Пользователь" (стр. 65)) для обновления в этой переменной:

```
>>> user_data = {
...     "contact": {
...         "address1": "John Doe Str.",
...         "firstname": "Foo",
...         "lastname": "Bar"
...     },
...     "version": version
... }
```

Имя	Тип значения	Обязательно	Описание
contact	объект контактов (см. "Пользователь" (стр. 65))	Нет	Контактная информация учетной записи.
version	число	Да	Номер редакции.

5. Преобразуйте объект user_data в текст в формате JSON:

```
>>> user_data = json.dumps(user_data, indent=4)
```

6. Отправьте запрос PUT с текстом в формате JSON на конечную точку /users/<user_id>:

```
>>> response = requests.put(
...     f'{base_url}/users/{user_id}',
...     headers={'Content-Type': 'application/json', **auth},
...     data=user_data,
... )
```

7. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что контактные данные были успешно изменены.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, текст ответа содержит информацию об учетной записи пользователя (см. "Пользователь" (стр. 65)) в формате JSON. При преобразовании в объект это будет выглядеть следующим образом:

```
>>> pprint.pprint(response.json())
{'activated': False,
 'business_types': [],
 'contact': {
        'address1': 'John Doe Str.',
        'address2': '',
        'city': '',
        'country': '',
        'email': 'johndoe@mysite.com',
        'firstname': 'Foo',
        'lastname': 'Bar',
        'phone': '',
        'state': '',
        'zipcode': '',
        'id': '6e6d758d-8e74-3ae3-ac84-50eb0dff12eb',
        'types': [],
        'title': '',
        'website': '',
        'industry': '',
        'email_confirmed": false,
        'language': '',
        'fax': '',
        'created_at': '2019-07-25T07:11:02.807354+00:00',
        'updated_at': '2019-07-30T07:15:09.604233+00:00'
},
 'created_at': '2019-07-25T07:11:02.807354+00:00',
 'updated_at': '2019-07-30T07:15:09.604233+00:00',
 'deleted_at': None,
 'enabled': True,
 'id': '1c234e69-5469-424a-a6d1-ff5658b387a6',
 'language': 'en',
 'login': 'JohnDoe',
 'personal_tenant_id': None,
 'tenant_id': 'ede9f834-70b3-476c-83d9-736f9f8c7dae',
 'version': 2}
```

Номер редакции увеличивается с каждым обновлением. При необходимости обновите переменную version.

2.10.4.13 Удаление пользователя

Предупреждение

При удалении учетной записи пользователя все данные, связанные с этой учетной записью, будут удалены. Эта операция необратима.

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

2. Присвойте переменной user_id UUID учетной записи пользователя, созданной с помощью API (см. "Создание учетной записи пользователя" (стр. 70)), или учетной записи пользователя, найденной с помощью поиска (см. "Поиск пользователя" (стр. 74)):

   ```
   >>> user_id = created_user_id
   >>> user_id
   '1c234e69-5469-424a-a6d1-ff5658b387a6'
   ```

3. Отключите учетную запись пользователя. Для этого нужно установить значение атрибута enabled = False. Подробнее описано в статье "Изменение свойств пользователя" (стр. 80).
4. Извлеките номер редакции учетной записи пользователя, как описано в предыдущих главах. Следующая переменная должна быть доступна уже сейчас:

   ```
   >>> version
   1
   ```

5. Задайте переменную params, а затем присвойте этой переменной параметр version строки запроса:

   ```
   >>> params = {
   ...     'version': version
   ... }
   ```

Имя	Тип значения	Обязательно	Описание
version	число	Да	Номер редакции.

6. Отправьте запрос DELETE на конечную точку /users/<user_id>:

   ```
   >>> response = requests.delete(f'{base_url}/users/{user_id}', headers=auth, params=params)
   ```

7. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   204
   ```

   Код состояния HTTP 204 означает, что учетная запись пользователя была удалена.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

2.10.5 Политика доступа

Политика доступа – это набор правил, определяющих, к каким данным и функциям системы пользователь имеет доступ и с какими ограничениями он при этом сталкивается. Она определяется набором ролей в свойствах пользователя, которые возможно назначить пользователю в соответствии с ролевой моделью системы.

Все операции с **политиками доступа** в облачной платформе находятся в конечной точке /access_policies.

2.10.5.1 Структура объекта JSON политики доступа

Описание структуры объекта JSON политики доступа доступно по ссылке.

Имя	Тип значения	Описание
id	строка UUID	Уникальный идентификатор (UUID) политики доступа.
version	число	Номер редакции.
created_at	строка	Дата и время создания политики доступа согласно стандарту ISO 8601.
updated_at	строка	Дата и время последнего обновления политики доступа согласно стандарту ISO 8601.
deleted_at	строка	Дата и время мягкого удаления политики доступа согласно стандарту ISO 8601 или null в случае, если политика доступа не удалена.
trustee_id	строка UUID	Уникальный идентификатор (UUID) субъекта, для которого предоставляется политика доступа.
trustee_type	строка	Тип субъекта, для которого предоставляется политика доступа. Возможные значения: • USER • USER_GROUP • CLIENT
issuer_id	строка UUID	Уникальный идентификатор (UUID) назначающего политику доступа.
tenant_id	строка UUID	Уникальный идентификатор (UUID) тенанта владельца ресурса.
role_id	строка	Идентификатор роли.

2.10.5.2 Перечень ролей

ID	Описание
root_admin	Эта роль предоставляет права администратора для всех служб системы.
partner_admin	Эта роль предоставляет права администратора для всех служб тенанта партнёра, к которому относится пользователь: • портал управления; • защита; • Кибер Инфраструктура.
company_admin	Эта роль предоставляет права администратора для следующих служб клиента, к которому относится пользователь: • портал управления; • защита.
unit_admin	Эта роль предоставляет права администратора для следующих служб отдела компании клиента, к которому относится пользователь: • портал управления; • защита.
readonly_admin	Эта роль предоставляет права администратора с доступом только для чтения для службы "портал управления" тенанта (партнёра, клиента, отдела), к которому относится пользователь.
protection_admin	Эта роль предоставляет права администратора для службы "защита" тенанта (партнёра, клиента, отдела), к которому относится пользователь.
protection_ro_admin	Эта роль предоставляет права администратора с доступом только для чтения для службы "защита" тенанта (партнёра, клиента, отдела), к которому относится пользователь.
restore_operator	Эта роль предоставляет ограниченный доступ к функции восстановления резервных копий.
backup_user	Эта роль предоставляет права пользователя для службы "защита" тенанта (клиента, отдела), к которому относится пользователь.
hci_admin	Эта роль предоставляет права администратора для службы "Кибер Инфраструктура" тенанта партнёра, к которому относится пользователь.

2.10.5.3 Пример политики доступа

```
{
        "id": "00000000-0000-0000-0000-00000002aaa9",
        "version": 0,
        "trustee_id": "c602f559-f11b-4c86-af6d-6b4d5cc222b6",
        "trustee_type": "USER",
```

```
        "issuer_id": "901c0ea0-1523-4e76-b052-4e185d21ecf1",
        "tenant_id": "901c0ea0-1523-4e76-b052-4e185d21ecf1",
        "role_id": "root_admin",
        "created_at": "2025-04-01T06:13:47Z",
        "updated_at": "2025-04-01T06:13:47Z",
        "deleted_at": null
}
```


2.10.5.4 Справочники

Справочники доступны по ссылке.

2.10.5.5 Действия с политиками доступа

Операция	Используемые методы и конечные точки
"Получение ролей пользователя" (стр. 87)	GET /access_policies?user_id=<user ID>
"Изменение роли пользователя" (стр. 88)	POST /access_policies

2.10.5.6 Получение ролей пользователя

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).
2. Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   >>> tenant_id  # the UUID of the tenant to which the token provides access
   'ede9f834-70b3-476c-83d9-736f9f8c7dae'
   ```

3. Присвойте переменной user_id UUID учетной записи пользователя, созданной с помощью API (см. "Создание учетной записи пользователя" (стр. 70)), или учетной записи пользователя, найденной с помощью поиска (см. "Поиск пользователя" (стр. 74)):

   ```
   >>> user_id = created_user_id
   >>> user_id
   '404fe9bf-3f8b-4618-b158-2ca526e6eafd'
   ```

4. Отправьте запрос GET на конечную точку /access_policies?user_id=<user_id>:

   ```
   >>> response = requests.get(f'{base_url}/access_policies?user_id={user_id}', headers=auth)
   ```

5. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   200
   ```

Код состояния HTTP 200 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, тело ответа содержит информацию об учетной записи пользователя в формате JSON. При преобразовании в объект это будет выглядеть следующим образом:

```
>>> pprint.pprint(response.json())

{
  "items": [
    {
      "id": "00000000-0000-0000-0000-00000002d837",
      "version": 0,
      "trustee_id": "404fe9bf-3f8b-4618-b158-2ca526e6eafd",
      "trustee_type": "USER",
      "issuer_id": "901c0ea0-1523-4e76-b052-4e185d21ecf1",
      "tenant_id": "901c0ea0-1523-4e76-b052-4e185d21ecf1",
      "role_id": "accounts_admin",
      "created_at": "2025-04-07T11:09:03Z",
      "updated_at": "2025-04-07T11:09:03Z",
      "deleted_at": null
    }
  ]
}
```

2.10.5.7 Изменение роли пользователя

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).
2. Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
>>> tenant_id  # the UUID of the tenant to which the token provides access
'ede9f834-70b3-476c-83d9-736f9f8c7dae'
```

3. Присвойте переменной user_id UUID учетной записи пользователя, созданной с помощью API (см. "Создание учетной записи пользователя" (стр. 70)), или учетной записи пользователя, найденной с помощью поиска (см. "Поиск пользователя" (стр. 74)):

```
>>> user_id = created_user_id
>>> user_id
'404fe9bf-3f8b-4618-b158-2ca526e6eafd'
```

4. Получите информацию о текущей роли пользователя (см. "Получение ролей пользователя" (стр. 87)).

5. Задайте переменную user_data, а затем присвойте этой переменной полученную информацию о роли пользователя со следующими изменениями:
   - добавьте объект на каждую роль, которая будет у пользователя;
   - значение атрибута id должно быть 00000000-0000-0000-0000-000000000000
   - значения атрибута role_id (перечень ролей описан в разделе "Политика доступа" (стр. 85));

```
>>> user_data = {
..."items": [
...    {
...    "id": "00000000-0000-0000-0000-000000000000",
...    "issuer_id": "901c0ea0-1523-4e76-b052-4e185d21ecf1",
...    "role_id": "accounts_ro_admin",
...    "tenant_id": "901c0ea0-1523-4e76-b052-4e185d21ecf1",
...    "trustee_id": "404fe9bf-3f8b-4618-b158-2ca526e6eafd",
...    "trustee_type": "USER",
...    "version": 0
...    },
...    {
...    "id": "00000000-0000-0000-0000-000000000000",
...    "issuer_id": "901c0ea0-1523-4e76-b052-4e185d21ecf1",
...    "role_id": "protection_ro_admin",
...    "tenant_id": "901c0ea0-1523-4e76-b052-4e185d21ecf1",
...    "trustee_id": "404fe9bf-3f8b-4618-b158-2ca526e6eafd",
...    "trustee_type": "USER",
...    "version": 0
...    },
...    {
...    "id": "00000000-0000-0000-0000-000000000000",
...    "issuer_id": "901c0ea0-1523-4e76-b052-4e185d21ecf1",
...    "role_id": "hci_admin",
...    "tenant_id": "901c0ea0-1523-4e76-b052-4e185d21ecf1",
...    "trustee_id": "404fe9bf-3f8b-4618-b158-2ca526e6eafd",
...    "trustee_type": "USER",
...    "version": 0
...    }
...]}
```

6. Преобразуйте объект user_data в текст в формате JSON:

```
>>> user_data = json.dumps(user_data, indent=4)
```

7. Отправьте запрос POST на конечную точку /access_policies:

```
>>> response = requests.post(
...    f'{base_url}/access_policies',
...    headers={'Content-Type': 'application/json', **auth},
...    data=user_data,
... )
```

8. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, тело ответа содержит информацию об учетной записи пользователя в формате JSON. При преобразовании в объект это будет выглядеть следующим образом:

```
>>> pprint.pprint(response.json())

{
   'items': [
      {
         'id': '00000000-0000-0000-0000-00000002d83a',
         'version': 0,
         'trustee_id': '404fe9bf-3f8b-4618-b158-2ca526e6eafd',
         'trustee_type': 'USER',
         'issuer_id': '901c0ea0-1523-4e76-b052-4e185d21ecf1',
         'tenant_id': '901c0ea0-1523-4e76-b052-4e185d21ecf1',
         'role_id': 'accounts_ro_admin',
         'created_at': '2025-04-08T12:48:56Z',
         'updated_at': '2025-04-08T12:50:07Z',
         'deleted_at': null
      },
      {
         'id': '00000000-0000-0000-0000-00000002d83e',
         'version': 0,
         'trustee_id': '404fe9bf-3f8b-4618-b158-2ca526e6eafd',
         'trustee_type': 'USER',
         'issuer_id': '901c0ea0-1523-4e76-b052-4e185d21ecf1',
         'tenant_id': '901c0ea0-1523-4e76-b052-4e185d21ecf1',
         'role_id': 'protection_ro_admin',
         'created_at': '2025-04-08T12:45:15Z',
         'updated_at': '2025-04-08T12:48:56Z',
         'deleted_at': null,
      },
      {
         'id': '00000000-0000-0000-0000-00000002d838',
         'version': 0,
         'trustee_id': '404fe9bf-3f8b-4618-b158-2ca526e6eafd',
         'trustee_type': 'USER',
         'issuer_id': '901c0ea0-1523-4e76-b052-4e185d21ecf1',
         'tenant_id': '901c0ea0-1523-4e76-b052-4e185d21ecf1',
         'role_id': 'hci_admin',
         'created_at': '2025-04-08T12:45:15Z',
         'updated_at': '2025-04-08T12:48:56Z',
         'deleted_at': null,
      }
   ]
}
```

2.10.6 Ресурс

Ресурс (также известный как рабочая нагрузка) — это источник данных, такой как физические и виртуальные хосты, базы данных, почтовые ящики, веб-сайты и т.д., к которому можно применить план защиты.

Ресурсы служат контекстами, к которым применяются и для которых выполняются правила или параметры политики.

Ресурс может быть атомарным или групповым. Атомарный ресурс может входить в несколько групповых ресурсов. Групповой ресурс можно определить статически (явно задав его участие в группе) или динамически (с помощью набора запросов к ресурсам).

Все операции с ресурсами находятся в конечной точке /resources.

2.10.6.1 Структура объекта JSON ресурса

Описание структуры объекта JSON ресурса доступно по ссылке.

Имя	Тип значения	Описание
id	строка UUID	Уникальный идентификатор (UUID) ресурса.
created_at	строка	Дата и время создания ресурса согласно стандарту ISO 8601.
updated_at	строка	Дата и время последнего обновления ресурса согласно стандарту ISO 8601.
deleted_at	строка	Дата и время мягкого удаления (установки флага "удалён") ресурса согласно стандарту ISO 8601 или null в случае, если ресурс не удалён.
tenant_id	строка UUID	Уникальный идентификатор (UUID) тенанта владельца ресурса.
external_id	строка UUID	Уникальный идентификатор (UUID) ресурса из внешней системы.
type	строка	Тип ресурса в формате type.scope, где type — это resource, a scope может принимать следующие значения: • machine • datastore.vmwesx • datastore_cluster.vmwesx • virtual_appliance.vmwesx • virtual_application.vmwesx • virtual_center.vmwesx • virtual_cluster.vmwesx • virtual_data_center.vmwesx • virtual_folder.vmwesx • virtual_machine.vmwesx • virtual_network.vmwesx

Имя	Тип значения	Описание
		• virtual_resource_pool.vmwesx • virtual_host.vmwesx • virtual_machine.vmww • virtual_cluster.mshyperv • virtual_machine.mshyperv • virtual_host.mshyperv • virtual_appliance.mshyperv • virtual_network.mshyperv • virtual_folder.mshyperv • virtual_data_center.mshyperv • datastore.mshyperv • virtual_machine.parallelsw • virtual_host.parallelsw • virtual_cluster.pcs • virtual_machine.msvpc • virtual_machine.msvs • virtual_machine.pcs • virtual_machine.xen • virtual_machine.kvm • virtual_machine.kvmsrv • virtual_machine.rhev • virtual_host.vmww • virtual_host.msvpc • virtual_host.msvs • virtual_host.pcs • virtual_host.xen • virtual_host.kvm • virtual_host.kvmsrv • virtual_host.rhev • virtual_machine.hci • virtual_host.hci • virtual_cluster.hci • virtual_machine.scale • virtual_host.scale • virtual_cluster.scale • group.computers • group.all • group.vcenter • group.datacenter • group.hosts_and_clusters • group.cluster

Имя	Тип значения	Описание
		• group.host • group.virtual_application • group.resource_pool • group.virtual_folder • group.vms_and_templates • group.vms_folder • mssql_server • mssql_aag_database • mssql_aag_group • mssql_database • mssql_database_folder • mssql_instance • mssql_system_database • exchange • msexchange_database • msexchange_storage_group • msexchange_mailbox.msexchange • msexchange_mailbox.office365
settings_schema	строка	Версия схемы настроек.
agent_id	строка	Идентификатор агента.
group_condition	строка	Список ресурсов, используемых внутри группы при её создании. Параметр должен включать: • для группы доменов – тип ресурса, домен, экземпляр; • для группы экземпляров – тип ресурса и экземпляр.
allowed_member_types	массив строк	Список типов ресурсов, которые можно добавить в группу.
parent_group_ids	массив строк UUID	Перечень уникальных идентификаторов (UUID) родительских групп.
name	строка	Наименование ресурса.
user_defined_name	строка	Наименование ресурса, определённое пользователем.
tag	строка	Определённое пользователем строковое значение для фильтрации ресурсов.
attributes	объект	Массив атрибутов ресурса.

Имя	Тип значения	Описание
		Доступные базовые пространства имён: • default • legacy • tenant • agent • cyberfit • backups • hwi • workload_isolation

2.10.6.2 Структура объекта attributes

Описание атрибутов пространства имён доступно по ссылке.

Имя	Тип значения	Описание
name	строка	Наименование атрибута (пространство имён).
kvs	массив объектов	Массив объектов вида "ключ": "значение".

2.10.6.3 Структура объекта kvs

Имя	Тип значения	Описание
key	строка	Наименование ключа.
value	строка или число, массив строк или чисел	Единичное значение или массив из строк или целых чисел.

2.10.6.4 Структура объекта details

Имя	Тип значения	Описание
key	строка	Наименование ключа.
value	строка или число, массив строк или чисел	Единичное значение, представляющее собой строку или целое число, или массив, состоящий из строк или целых чисел.

2.10.6.5 Пример ресурса


```
{
  "id":"054b939a-3627-41c9-a351-151be3de0980",
  "created_at":"2021-01-18T14:38:24.5589857Z",
```

```
      "updated_at":"2021-01-18T14:38:24.5789853Z",
      "tenant_id":"cbad0dcf-fe8f-4989-bc03-27ae36112104",
      "external_id":"054b939a-3627-41c9-a351-151be3de0980@cbad0dcf-fe8f-4989-bc03-
27ae36112104",
      "type":"resource.group.computers",
      "group_condition":"comment like 'Simple*'",
      "parent_group_ids":[
        "301d1574-849e-4714-859f-3a2ec12a218b"
      ],
      "allowed_member_types":[
        "resource.machine"
      ],
      "name":"Dynamic group",
      "user_defined_name":"Dynamic group",
      "attributes":[
        {
          "name":"group",
          "kvs":[
            {
              "key":"parent_id",
              "value":[
                "62E5834D-FB8E-500D-956E-02C182D463D2"
              ]
            },
            {
              "key":"tz_offset",
              "value":[
                "480"
              ]
            }
          ]
        }
      ]
    }
```


2.10.6.6 Справочники

Справочники доступны по ссылке.

2.10.6.7 Действия с ресурсами

Операция	Используемые методы и конечные точки
"Получение списка ресурсов" (стр. 96)	GET /resources
"Получение подробной информации о ресурсе" (стр. 96)	GET /resources/<resource_id>

2.10.6.8 Получение списка ресурсов

1. Выполните аутентификацию с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступными следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   ```

2. Отправьте запрос GET на конечную точку /resources:

   ```
   >>> response = requests.get(f'{base_url}/resources', headers=auth)
   ```

3. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   200
   ```

   Код состояния HTTP 200 означает, что запрос был успешно выполнен.

   Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

   Кроме того, тело ответа содержит объект с ключом items в формате JSON, содержащий массив объектов ресурсов. После преобразования в объект он будет выглядеть так:

   ```
   >>> pprint.pprint(response.json())
   {'items': [{'created_at': '2021-02-03T14:57:07.293995787Z',
           'external_id': '23effcf6-2798-4631-9a52-5785bf3af657@cbad0dcf-fe8f-4989-bc03-
   27ae36112104',
           'id': '23effcf6-2798-4631-9a52-5785bf3af657',
           'name': 'DESKTOP-JRPTA4A',
           'tenant_id': 'cbad0dcf-fe8f-4989-bc03-27ae36112104',
           'type': 'resource.machine',
           'updated_at': '2021-02-03T18:13:48.312293448Z',
           'user_defined_name': 'DESKTOP-JRPTA4A'},
           ...],
   'paging': {'cursors': {"after": ""}}}
   ```

4. Преобразуйте текст JSON, содержащийся в теле ответа, в объект, а затем извлеките список ресурсов из ответа:

   ```
   >>> resources = response.json()['items']
   ```

2.10.6.9 Получение подробной информации о ресурсе

1. Выполните аутентификацию с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
```

2. Получите список ресурсов, как описано в разделе "Получение списка ресурсов" (стр. 96). В качестве примера будет взят идентификатор первого ресурса. В результате должна быть доступна следующая переменная:

```
>>> resource_id = resources[0]['id']
>>> resource_id
'054b939a-3627-41c9-a351-151be3de0980'
```

3. Отправьте запрос GET на конечную точку /resources/{resource_id}:

```
>>> response = requests.get(f'{base_url}/resources/{resource_id}', headers=auth)
```

4. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

После преобразования в объект ресурс будет выглядеть так:

```
>>> pprint.pprint(response.json())
{
      'id':'054b939a-3627-41c9-a351-151be3de0980',
      'created_at':'2021-01-18T14:38:24.5589857Z',
      'updated_at':'2021-01-18T14:38:24.5789853Z',
      'tenant_id':'00000000-0000-0000-0000-000000000000',
      'external_id':'054b939a-3627-41c9-a351-151be3de0980@00000000-0000-0000-0000-
000000000000',
      'type':'resource.group.computers',
      'group_condition':'comment like 'Simple*'',
      'parent_group_ids':[
        '301d1574-849e-4714-859f-3a2ec12a218b'
      ],
      'allowed_member_types':[
         'resource.machine'
      ],
      'name':'Dynamic group',
      'user_defined_name':'Dynamic group',
      'attributes': [{'kvs': [{'key': 'components', 'value': ['HyperVAgent', 'x64WindowsAgent']},
                              {'key': 'features', 'value': ['HyperVAgent', 'x64WindowsAgent']},
                              {'key': 'os_family', 'value': 'Windows'},
                              {'key': 'os_name', 'value': 'Microsoft Windows Server 2019 Standard'},
```

```
                {'key': 'os_arch', 'value': 'x64'},
                {'key': 'os_family', 'value': 'Windows'},
                {'key': 'os_caps', 'value': 18},
                {'key': 'os_product_type', 'value': 3},
                {'key': 'os_service_pack', 'value': 0},
                {'key': 'os_suite_mask', 'value': 272},
                {'key': 'sku', 'value': 7},
                {'key': 'os_version_major', 'value': 10},
                {'key': 'os_version_minor', 'value': 0},
                {'key': 'mac_addresses', 'value': ['00:50:56:84:DB:B7']},
                {'key': 'memory_size', 'value': 17178800128},
                {'key': 'processor_frequency', 'value': 3504},
                {'key': 'processor_name', 'value': 3504},
                {'key': 'chassis', 'value': 'other'},
                {'key': 'vm_host_type', 'value': 'vmwesx'},
                {'key': 'esx_address', 'value': ''},
                {'key': 'residental_addresses', 'value': ['10.34.16.186']},
                {'key': 'inside_virtual', 'value': 1},
                {'key': 'tz_offset', 'value': 120},
                {'key': 'version', 'value': '15.0.26355'},
                {'key': 'enabled', 'value': 1},
                {'key': 'is_online', 'value': 0}],
            'name': 'agent'},
            {'details':   [{'key': 'cyberfit_score_details',
                    'value': {'bitlocker': {'encrypted_volumes': 0,
                                            'is_system_volume_encrypted': False,
                                            'volumes': 2},
                              'os': {'name': 'Microsoft Windows Server 2019 Standard',
                                     'version': '10.0.17763'},
                              'programs': {'antispyware': [{'name': 'Cyber Protect Agent',
                                                            'vendor': 'Cyberprotect',
                                                            'version': '15.0.26355'}],
                                           'antivirus': [{'name': 'Cyber Protect Agent',
                                                          'vendor': 'Cyberprotect',
                                                          'version': '15.0.26355'}],
                                           'backup': [{'name': 'Cyber Protect Agent',
                                                       'vendor': 'Cyberprotect',
                                                       'version': '15.0.26355'}],
                                           'disk_encryption': [],
                                           'firewall': [],
                                           'vpn': []},
                              'windows_defender': {'antispyware': {'enabled': True},
                                                   'antivirus': {'enabled': True}},
                              'windows_firewall': {'private_network': {'enabled': True},
                                                   'public_network': {'enabled': True}},
                              'windows_server_backup': {'installed': False}}},
                    {'key': 'cyberfit_score_value', 'value': 625},
                    {'key': 'cyberfit_score_server_version', 'value': 38},
                    {'key': 'cyberfit_score_client_version', 'value': 108},
                    {'key': 'cyberfit_score_assessment_date', 'value': '2021-02-
03T18:13:46.9931835Z'},
                    {'key': 'cyberfit_score_metrics',
```

```
                    'value': {'antimalware': {'current': 275, 'max': 275},
                         'backup': {'current': 175, 'max': 175},
                         'disk_encryption': {'current': 0,
                                             'max': 125},
                         'firewall': {'current': 175, 'max': 175},
                         'ntlm': {'current': 0, 'max': 25},
                         'vpn': {'current': 0, 'max': 75}}}],
      'kvs': [{'key': 'cyberfit_score_value', 'value': '625'}],
      'name': 'cyberfit'},
    {'kvs': [{'key': 'ip_addresses', 'value': '10.34.16.186'},
             {'key': 'ldap_status', 'value': 'ldapoff'},
             {'key': 'operating_system_product_type', 'value': 3},
             {'key': 'operating_system_type', 'value': 3},
             {'key': 'architecture', 'value': 'x64'},
             {'key': 'ip', 'value': '10.34.16.186'},
             {'key': 'name', 'value': 'DESKTOP-JRPTA4A'},
             {'key': 'operating_system', 'value': 'Microsoft Windows Server 2019 Standard'},
             {'key': 'os_product_type', 'value': '3'},
             {'key': 'residential_address_v4', 'value': '0a2210ba'}],
     'name': 'default'}]}
```


2.10.7 Группа

Группа – это набор ресурсов, к которому можно применить план защиты. Группы упрощают управление большим количеством рабочих нагрузок.

Группы подразделяются на типы:

- **Статические группы** содержат рабочие нагрузки, добавленные вручную.
- **Динамические группы** содержат рабочие нагрузки, добавленные автоматически в соответствии с поисковыми критериями, определенными при создании группы. Состав динамической группы меняется автоматически. Рабочая нагрузка остается в группе до тех пор, пока отвечает заданным критериям.

После появления рабочей нагрузки в группе, она будет защищена планом. Если рабочая нагрузка удалена из группы, она больше не будет защищена планом.

Все операции с группами находятся в конечной точке /groups.

2.10.7.1 Структура объекта JSON группы

Описание структуры объекта JSON группы доступно по ссылке.

2.10.7.2 Справочники

Справочники доступны по ссылке.

2.10.7.3 Действия с группами

Операция	Используемые методы и конечные точки
"Создание группы" (стр. 100)	POST /groups
"Обновление группы" (стр. 101)	PUT /groups/<group_id>
"Удаление группы" (стр. 103)	DELETE /groups/<group_id>
"Добавление ресурсов в статическую группу или удаление их из нее" (стр. 103)	PATCH /groups/<resource_id>/resources

2.10.7.4 Создание группы

1. Выполните аутентификацию с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   ```

2. Запросите ресурсы, как описано в разделе "Получение списка ресурсов" (стр. 96). Задайте переменную parent_id и присвойте ей в качестве значения parent_id ресурса с параметром "name": "All":

   ```
   >>> parent_id = next(
      (item['parent_id'] for item in data['items'] if item['name'] == "All"),
      None
   )
   >>> parent_id
   301D1574-849E-4714-859F-3A2EC12A218B
   ```

3. Задайте переменную group_data, а затем присвойте этой переменной объект, содержащий параметры создаваемой группы:

   ```
   >>> group_data = {
    'name': 'Static group',
    'membership_type': 'STATIC',
    'parent_id': parent_id,
    'type': 'group.computers',
    'comment': 'Comment static group'
   }
   ```

Примечание

Условия формирования динамической группы передаются в параметре condition.

Доступные параметры и правила формирования условий описаны в данном разделе документации.

4. Отправьте запрос POST на конечную точку /groups:

```
>>> response = requests.post(
...     f'{base_url}/groups',
...     headers={'Content-Type': 'application/json', **auth},
...     data=group_data,
... )
```

5. Проверьте код состояния запроса:

```
>>> response.status_code
201
```

Код состояния HTTP 201 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

После преобразования в объект группа будет выглядеть так:

```
>>> pprint.pprint(response.json())
{
  'comment': 'Comment static group',
  'created_at': '2026-03-03T14:45:36.967158383Z',
  'id': '5AFCBCF6-6070-47A9-9569-69C8D3E68CE5',
  'member_types': [
    'machine'
  ],
  'membership_type': 'STATIC',
  'name': 'Static group',
  'parent_id': '301D1574-849E-4714-859F-3A2EC12A218B',
  'tenant_id': '82632930-5a7c-45c1-aec0-f0e8a37ef414',
  'type': 'group.computers',
  'updated_at': '2026-03-03T14:45:36.967158383Z'
}
```

2.10.7.5 Обновление группы

1. Выполните аутентификацию с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

Должны стать доступными следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
```

```
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
```

2. Запросите ресурсы, как описано в разделе "Получение списка ресурсов" (стр. 96). Задайте переменную group_id нужной группы и присвойте ей в качестве значения id ресурса. В примере использован ресурс с параметром "name": "Static group":

```
>>> group_id = next(
   (item['id'] for item in data['items'] if item['name'] == "Static group"),
   None
)
>>> group_id
5AFCBCF6-6070-47A9-9569-69C8D3E68CE5
```

3. Задайте переменную group_data, а затем присвойте этой переменной объект, содержащий параметры создаваемой группы:

```
>>> group_data = {
  'name': 'Static group 2',
  'comment': 'Новый комментарий для группы'
}
```

4. Отправьте запрос PUT на конечную точку /groups/{group_id}:

```
>>> response = requests.put(
...     f'{base_url}/groups/{group_id}',
...     headers={'Content-Type': 'application/json', **auth},
...     data=group_data,
... )
```

5. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

После преобразования в объект группа будет выглядеть так:

```
>>> pprint.pprint(response.json())
{
  'comment': 'Новый комментарий для группы',
  'created_at': '2026-03-03T14:45:36.967158383Z',
  'id': '5AFCBCF6-6070-47A9-9569-69C8D3E68CE5',
  'member_types': [
    'machine'
  ],
  'membership_type': 'STATIC',
```

```
  'name': 'Static group 2',
  'parent_id': '301D1574-849E-4714-859F-3A2EC12A218B',
  'tenant_id': '82632930-5a7c-45c1-aec0-f0e8a37ef414',
  'type': 'group.computers',
  'updated_at': '2026-03-03T14:45:36.967158383Z'
}
```

2.10.7.6 Удаление группы

1. Выполните аутентификацию с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступными следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   ```

2. Запросите ресурсы, как описано в разделе "Получение списка ресурсов" (стр. 96). Задайте переменную group_id нужной группы и присвойте ей в качестве значения id ресурса. В примере использован ресурс с параметром "name": "Static group":

   ```
   >>> group_id = next(
      (item['id'] for item in data['items'] if item['name'] == "Static group"),
      None
   )
   >>> group_id
   5AFCBCF6-6070-47A9-9569-69C8D3E68CE5
   ```

3. Отправьте запрос DELETE на конечную точку /groups/{group_id}:

   ```
   >>> requests.delete(f'{base_url}/groups/{group_id}', headers=auth)
   ```

4. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   204
   ```

   Код состояния HTTP 204 означает, что запрос был успешно выполнен.

   Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

2.10.7.7 Добавление ресурсов в статическую группу или удаление их из нее

1. Выполните аутентификацию с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступными следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
```

2. Запросите ресурсы, как описано в разделе "Получение списка ресурсов" (стр. 96). Задайте переменную group_id нужной группы и присвойте ей в качестве значения id ресурса. В примере использован ресурс с параметром "name": "Static group":

```
>>> group_id = next(
   (item['id'] for item in data['items'] if item['name'] == "Static group"),
   None
)
>>> group_id
5AFCBCF6-6070-47A9-9569-69C8D3E68CE5
```

3. В примере будем добавлять ресурсы в группу. Задайте переменную group_data, а затем присвойте этой переменной объект, содержащий перечень ресурсов, которые нужно добавить в группу:

```
>>> group_data = {
 'add': [
  '301D1574-849E-4714-859F-3A218A2EC12B',
  '301D1574-849E-4714-859F-3A2A21EC128B'
 ]
}
```

4. Отправьте запрос PATCH на конечную точку /groups/{group_id}/resources:

```
>>> response = requests.patch(
...    f'{base_url}/groups/{group_id}/resources',
...    headers={'Content-Type': 'application/json', **auth},
...    data=group_data,
... )
```

5. Проверьте код состояния запроса:

```
>>> response.status_code
204
```

   Код состояния HTTP 204 означает, что запрос был успешно выполнен.

   Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

2.10.8 Политика

Политика (план резервного копирования) – это набор настроек для обеспечения защиты данных, который можно применить к ресурсам и запустить с помощью агента защиты. Политика используется для настройки резервного копирования и восстановления данных.

Все операции с политиками находятся в конечной точке /policies.

2.10.8.1 Типы политик

Префикс имени	Описание
policy.backup	Политика резервного копирования.
policy.recovery	Политика восстановления данных (запланировано в следующих версиях документации открытого API).

2.10.8.2 Структура объекта JSON политики

Описание структуры объекта JSON политики доступно по ссылке.

Имя	Тип значения	Описание
id	строка UUID	Уникальный идентификатор (UUID) политики.
created_at	строка	Дата и время создания политики согласно стандарту ISO 8601.
updated_at	строка	Дата и время последнего обновления политики согласно стандарту ISO 8601.
deleted_at	строка	Дата и время мягкого удаления (установки флага "удалён") политики согласно стандарту ISO 8601 или null в случае, если ресурс не удалён.
tenant_id	строка UUID	Уникальный идентификатор (UUID) тенанта владельца политики.
type	строка	Тип политики. Наименования типов политик, доступных в текущей версии API, описаны в разделе "Справочник по настройкам политик" (стр. 172).
enabled	логическое значение	Флаг, который отключает или включает политику на платформе.
name	строка	Название политики.
settings	объект	Описание параметров настроек политик, доступных в текущей версии API, представлено в разделе "Справочник по настройкам политик" (стр. 172).
settings_ constraints	строка, число, логическое значение или объект	Ограничивающие правила для текущих параметров политики. Допустимые значения: диапазоны чисел, маски строковых значений и др.
validation_ dependencies	массив строк UUID	Ссылки на зависимости. Допускаются только уникальные идентификаторы (UUID) дочерних политик в рамках одного составного элемента (иные идентификаторы игнорируются).

Имя	Тип значения	Описание
		Должен заполняться на первом этапе проверки политики.
ls_features	строка	Перечень лицензий, рассчитанный на основе полей "settings". Не заполняется, когда поле "enabled" имеет значение false.

2.10.8.3 Пример политики

Примеры политик доступны в разделе "Параметры политики защиты" (стр. 106).

2.10.8.4 Справочники

Справочники доступны по ссылке.

2.10.8.5 Действия с политиками

Операция	Используемые методы и конечные точки
"Создание плана резервного копирования" (стр. 125)	POST /policies
"Получение перечня планов резервного копирования" (стр. 128)	GET /policies
"Получение информации о плане резервного копирования" (стр. 137)	GET /policies/{policy_id}
"Редактирование параметров плана резервного копирования" (стр. 141)	PATCH /policies/{policy_id}
"Включение/выключение плана резервного копирования" (стр. 142)	PATCH /policy_applications/enabled
"Удаление плана резервного копирования" (стр. 144)	DELETE /policies/{policy_id}

2.10.8.6 Параметры политики защиты

Политика резервного копирования машины

Политика обеспечивает функциональность резервного копирования рабочей нагрузки машины целиком.

Дополнительные сведения о параметрах планов резервного копирования см. в данном разделе пользовательской документации по продукту Кибер Бэкап Облачный.

Описание параметров запроса для создания политики резервного копирования см. данном разделе справочника API.

Следующий пример можно использовать при создании планов защиты с данной политикой:

```
{
  'enabled': true,
  'ls_features': ' ',
  'name': 'физическая машина. резервная копия всей машины',
  'settings': {
    'alerts': {
      'enabled': false,
      'max_days_without_backup': 0
    },
    'archive': {
      'compression': 'MAX',
      'encryption': {
        'algorithm': 'AES192',
        'password': 'string'
      },
      'format': '12',
      'name': '[Machine Name]-[Plan ID]-[Unique ID]A[Plan ID][Plan name][Unique ID][Virtualization
Server Type]F',
      'splitting': {
        'size': 214748364800
      }
    },
    'cbt': 'USE_IF_ENABLED',
    'check_file_timestamp': true,
    'destination': {
      'location': {
        'settings': {
          'name': 'customer-dg09',
          'storage_type': 'LOCAL_FOLDER',
          'uri': 'E:/',
          'use_policy_credentials': true
        },
        'type': 'VAULT'
      },
      'performance_window': {
        'enabled': false
      },
      'retention': {
        'execution': 'AFTER_BACKUP',
        'rules': [
          {
            'max_count': 7
          }
        ]
      }
    },
    'error_handling': {
      'enabled': true,
      'interval': {
        'count': 1,
        'type': 'MINUTES'
      },
```

```
    'max_attempts': 1
  },
  'exclude_asz': true,
  'exclude_page_files': true,
  'fast_backup_enabled': true,
  'file_filters': {
    'exclude_hidden': true
  },
  'ignore_bad_sectors': true,
  'lvm_snapshotting_enabled': true,
  'multi_volume_snapshotting_enabled': true,
  'notarization': {
    'enabled': true
  },
  'owner_id': '32e95e91-39f4-6053-9ce4-0da7e86c8b81',
  'preserve_file_security_settings': true,
  'reset_free_blocks_list': true,
  'scheduling': {
    'backup_sets': [
      {
        'schedule': {
          'alarms': {
            'time': {
              'rand_max_delay': {
                'count': 30,
                'type': 'MINUTES'
              },
              'repeat_at': [
                {
                  'hour': 9,
                  'minute': 0
                }
              ],
              'weekdays': [
                'MON',
                'TUE',
                'WED'
              ]
            }
          },
          'conditions': {},
          'prevent_sleep': true,
          'type': 'DAILY'
        },
        'type': 'AUTO'
      }
    ],
    'enabled': true,
    'rand_max_delay': {
      'count': 30,
      'type': 'MINUTES'
    },
```

```
        'scheme': 'WEEKLY_FULL_DAILY_INCREMENTAL',
        'weekly_backup_day': 'MON'
      },
      'sector_by_sector': true,
      'silent_mode_enabled': true,
      'tapes': {
        'devices': null,
        'eject_media_if_successfull': false,
        'multiplexing': {
          'enabled': false,
          'streams_per_drive': 0
        },
        'multistreaming': {
          'enabled': false
        },
        'overwrite_data_on_tape': false,
        'preserve_tapes_position': true,
        'tapeset': ''
      },
      'validate_backup': true,
      'vss': {
        'enabled': true,
        'provider': 'NATIVE'
      }
    },
    'tenant_id': '00000000-0000-0000-0000-000000000000',
    'type': 'policy.backup.machine'
}
```

Политика резервного копирования диска

Политика обеспечивает функциональность резервного копирования рабочей нагрузки диска или тома машины.

Дополнительные сведения о параметрах планов резервного копирования см. в данном разделе пользовательской документации по продукту Кибер Бэкап Облачный.

Описание параметров запроса для создания политики резервного копирования см. данном разделе справочника API.

Следующий пример можно использовать при создании планов защиты с данной политикой:

```
{
    'enabled': true,
    'ls_features': '',
    'name': 'виртуальная машина. резервная копия диска',
    'settings': {
      'alerts': {
        'enabled': false,
        'max_days_without_backup': 0
      },
```

```
'archive': {
  'compression': 'NORMAL',
  'format': 'AUTO',
  'name': '[Machine Name]-[Plan ID]-[Unique ID]A',
  'splitting': {
    'size': 214748364800
  }
},
'cbt': 'ENABLE_AND_USE',
'destination': {
  'location': {
    'settings': {
      'path': 'F:/local_auto_backups/'
    },
    'type': 'LOCAL_FOLDER_TEMPLATE'
  },
  'performance_window': {
    'enabled': false
  },
  'retention': {
    'execution': 'AFTER_BACKUP',
    'rules': [
      {
        'backup_set': 'DAILY',
        'max_age': {
          'count': 7,
          'type': 'DAYS'
        }
      },
      {
        'backup_set': 'WEEKLY',
        'max_age': {
          'count': 4,
          'type': 'WEEKS'
        }
      },
      {
        'backup_set': 'MONTHLY',
        'max_age': {
          'count': 6,
          'type': 'MONTHS'
        }
      }
    ]
  }
},
'error_handling': {
  'enabled': true,
  'interval': {
    'count': 30,
    'type': 'SECONDS'
  },
```

```
    'max_attempts': 300
  },
  'fast_backup_enabled': true,
  'file_filters': {},
  'fixed_drive_behavior_for_removable': true,
  'multi_volume_snapshotting_enabled': true,
  'notarization': {
    'enabled': false
  },
  'owner_id': 'daec7891-3a66-ed85-7aa6-2dbb0d6e0c4e',
  'preserve_file_security_settings': true,
  'quiesce_snapshotting_enabled': true,
  'replication': [
    {
      'location': {
        'settings': {
          'type': 'CYBERPROTECT'
        },
        'type': 'CLOUD_STORAGE'
      },
      'performance_window': {
        'enabled': false
      },
      'retention': {
        'execution': 'AFTER_BACKUP',
        'rules': [
          {
            'backup_set': 'DAILY',
            'max_age': {
              'count': 7,
              'type': 'DAYS'
            }
          },
          {
            'backup_set': 'WEEKLY',
            'max_age': {
              'count': 4,
              'type': 'WEEKS'
            }
          },
          {
            'backup_set': 'MONTHLY',
            'max_age': {
              'count': 6,
              'type': 'MONTHS'
            }
          }
        ]
      }
    }
  ],
  'scheduling': {
```

```
    'backup_sets': [
      {
        'schedule': {
          'alarms': {
            'time': {
              'monthdays': [
                1,
                15
              ],
              'months': [
                1,
                2,
                3,
                4,
                5,
                6,
                7,
                8,
                9,
                10,
                11,
                12
              ],
              'repeat_at': [
                {
                  'hour': 12,
                  'minute': 30
                }
              ]
            }
          },
          'conditions': {
            'users_idle': true
          },
          'prevent_sleep': true,
          'type': 'MONTHLY'
        },
        'type': 'FULL'
      }
    ],
    'enabled': true,
    'rand_max_delay': {
      'count': 30,
      'type': 'MINUTES'
    },
    'scheme': 'ALWAYS_FULL',
    'weekly_backup_day': 'MON'
  },
  'selection': {
    'items': [
      {
        'id': '[[\'LocalID\',\'\\\\\local\\\\hd_ev\\\\vol_guid
```

```
(8FE6E78C44C9D8F9B7B6A0950B209597)\'],[\'ItemType\',\'mms::disk::volume\'],[\'0B781614-
5AED-4A10-9B79-0A607CB7EEAE\',\'491168AF-257C-4FD2-8952-F76260B943C0\'],[\'4B2A7A93-
A44F-4155-BDE3-A023C57C9431\',\'gct::disks\']]',
                'name': 'C:'
            }
        ]
    },
    'silent_mode_enabled': true,
    'tapes': {
        'devices': null,
        'eject_media_if_successfull': false,
        'multiplexing': {
            'enabled': false,
            'streams_per_drive': 0
        },
        'multistreaming': {
            'enabled': false
        },
        'overwrite_data_on_tape': false,
        'preserve_tapes_position': true,
        'tapeset': ''
    },
    'vm_snapshot_error_handling': {
        'enabled': true,
        'interval': {
            'count': 5,
            'type': 'MINUTES'
        },
        'max_attempts': 3
    },
    'vss': {
        'enabled': true,
        'provider': 'TARGET_SYSTEM_DEFINED'
    }
  },
  'tenant_id': 'ca1adfdd-f0d4-4644-a22f-f6a1881d17dd',
  'type': 'policy.backup.disks'
}
```

Политика резервного копирования баз данных Microsoft SQL Server

Политика обеспечивает функциональность резервного копирования рабочей нагрузки машины целиком.

Дополнительные сведения о параметрах планов резервного копирования см. в данном разделе пользовательской документации по продукту Кибер Бэкап Облачный.

Описание параметров запроса для создания политики резервного копирования см. данном разделе справочника API.

Следующий пример можно использовать при создании планов защиты с данной политикой:

```
{
  "enabled": true,
  "ls_features": "",
  "name": "MS SQL. резервная копия баз данных",
  "settings": {
    "alerts": {
      "enabled": true,
      "max_days_without_backup": 5
    },
    "archive": {
      "comments": "",
      "compression": "HIGH",
      "display_name": "",
      "format": "AUTO",
      "name": "[Machine Name]-[Plan ID]-[Unique ID]A",
      "slice_comments": "",
      "splitting": {
        "auto": true
      }
    },
    "cluster": {
      "strategy": "REPLICA_FIRST"
    },
    "destination": {
      "location": {
        "settings": {
          "type": "CYBERPROTECT"
        },
        "type": "CLOUD_STORAGE"
      },
      "performance_window": {
        "enabled": false
      },
      "retention": {
        "execution": "AFTER_BACKUP",
        "rules": [
          {
            "backup_set": "DAILY",
            "max_age": {
              "count": 7,
              "type": "DAYS"
            }
          },
          {
            "backup_set": "WEEKLY",
            "max_age": {
              "count": 4,
              "type": "WEEKS"
            }
          }
        ]
      }
```

```
    },
    "error_handling": {
      "enabled": true,
      "interval": {
        "count": 30,
        "type": "SECONDS"
      },
      "max_attempts": 300,
      "silent_mode": true
    },
    "express_backup_usage": {
      "continue_files_tracking": true,
      "dont_create_session": false,
      "fallback_mode": "AUTO",
      "session_timeout": 0
    },
    "scheduling": {
      "backup_sets": [
        {
          "schedule": {
            "alarms": {
              "time": {
                "rand_max_delay": {
                  "count": 30,
                  "type": "MINUTES"
                },
                "repeat_at": [
                  {
                    "hour": 17,
                    "minute": 0
                  }
                ],
                "weekdays": [
                  "MON",
                  "TUE",
                  "WED",
                  "THU",
                  "FRI"
                ]
              }
            },
            "prevent_sleep": true,
            "type": "DAILY"
          },
          "type": "AUTO"
        }
      ],
      "enabled": true,
      "max_parallel_backups": 1,
      "rand_max_delay": {
        "count": 30,
        "type": "MINUTES"
```

```
        },
        "scheme": "WEEKLY_FULL_DAILY_INCREMENTAL",
        "task_failure": {
          "enabled": true,
          "interval": {
            "count": 1,
            "type": "HOURS"
          },
          "max_attempts": 5
        },
        "weekly_backup_day": "MON"
      },
      "tapes": {
        "eject_full_media": false,
        "eject_media_if_successfull": false,
        "multiplexing": {
          "enabled": false,
          "streams_per_drive": 0
        },
        "multistreaming": {
          "enabled": false
        },
        "overwrite_data_on_tape": false,
        "preserve_tapes_position": true,
        "tapeset": ""
      },
      "truncate_logs": true,
      "vss": {
        "enabled": true,
        "provider": "TARGET_SYSTEM_DEFINED"
      },
      "windows_event_log": {
        "enabled": true,
        "level": "ERRORS_AND_WARNINGS"
      }
    },
    "tenant_id": "f1579c67-f143-480b-9a13-095a345c6b0f",
    "type": "policy.backup.mssql"
}
```


Политика резервного копирования баз данных Microsoft Exchange Server

Политика обеспечивает функциональность резервного копирования рабочей нагрузки машины целиком.

Дополнительные сведения о параметрах планов резервного копирования см. в данном разделе пользовательской документации по продукту Кибер Бэкап Облачный.

Описание параметров запроса для создания политики резервного копирования см. данном разделе справочника API.

Следующий пример можно использовать при создании планов защиты с данной политикой:

```
{
  'enabled': true,
  'ls_features': '',
  'name': 'MS Exchange. резервная копия баз данных',
  'settings': {
    'alerts': {
      'enabled': false,
      'max_days_without_backup': 5
    },
    'archive': {
      'comments': '',
      'compression': 'HIGH',
      'display_name': '',
      'format': 'AUTO',
      'name': '[Database name]_[Unique ID]_[Plan ID]A',
      'slice_comments': '',
      'splitting': {
        'auto': true
      }
    },
    'cluster': {
      'strategy': 'REPLICA_FIRST'
    },
    'dag_list': [],
    'destination': {
      'location': {
        'settings': {
          'type': 'CYBERPROTECT'
        },
        'type': 'CLOUD_STORAGE'
      },
      'performance_window': {
        'enabled': false,
        'presets': [
          {
            'cpu_priority': 'LOW',
            'id': 'HIGH',
            'speed_limit': {
              'type': 'PERCENT',
              'value': 100
            },
            'timetable': [
              {
                'days_of_week': [
                  'MON',
                  'TUE',
                  'WED',
                  'THU',
                  'FRI',
                  'SAT',
                  'SUN'
                ],
```

```
                    'time_from': {
                      'hour': 0,
                      'minute': 0
                    },
                    'time_to': {
                      'hour': 23,
                      'minute': 59,
                      'second': 59
                    }
                  }
                ]
              },
              {
                'cpu_priority': 'LOW',
                'id': 'LOW',
                'speed_limit': {
                  'type': 'PERCENT',
                  'value': 25
                },
                'timetable': []
              },
              {
                'cpu_priority': 'NORMAL',
                'id': 'CANCEL',
                'speed_limit': {
                  'type': 'PERCENT',
                  'value': 100
                },
                'timetable': []
              }
            ]
          },
          'retention': {
            'execution': 'AFTER_BACKUP',
            'rules': [
              {
                'backup_set': 'DAILY',
                'max_age': {
                  'count': 7,
                  'type': 'DAYS'
                }
              },
              {
                'backup_set': 'WEEKLY',
                'max_age': {
                  'count': 4,
                  'type': 'WEEKS'
                }
              }
            ]
          }
        },
```

```
'error_handling': {
   'enabled': true,
   'interval': {
      'count': 30,
      'type': 'SECONDS'
   },
   'max_attempts': 300,
   'silent_mode': true
},
'express_backup_usage': {
   'continue_files_tracking': true,
   'dont_create_session': false,
   'fallback_mode': 'AUTO',
   'session_timeout': 0
},
'metadata_to_include': 'MAILBOXES',
'scheduling': {
   'backup_sets': [
      {
         'schedule': {
            'alarms': {
               'time': {
                  'rand_max_delay': {
                     'count': 30,
                     'type': 'MINUTES'
                  },
                  'repeat_at': [
                     {
                        'hour': 20,
                        'minute': 0
                     }
                  ],
                  'weekdays': [
                     'MON',
                     'THU',
                     'SAT'
                  ]
               }
            },
            'prevent_sleep': true,
            'type': 'DAILY'
         },
         'type': 'AUTO'
      }
   ],
   'enabled': true,
   'max_parallel_backups': 2,
   'rand_max_delay': {
      'count': 30,
      'type': 'MINUTES'
   },
   'scheme': 'WEEKLY_FULL_DAILY_INCREMENTAL',
```

```
            'task_failure': {
               'enabled': true,
               'interval': {
                  'count': 1,
                  'type': 'HOURS'
               },
               'max_attempts': 3
            },
            'weekly_backup_day': 'MON'
         },
         'tapes': {
            'eject_full_media': false,
            'eject_media_if_successfull': false,
            'multiplexing': {
               'enabled': false,
               'streams_per_drive': 0
            },
            'multistreaming': {
               'enabled': false
            },
            'overwrite_data_on_tape': false,
            'preserve_tapes_position': true,
            'tapeset': ''
         },
         'vss': {
            'enabled': true,
            'provider': 'TARGET_SYSTEM_DEFINED'
         },
         'windows_event_log': {
            'enabled': false,
            'level': 'ERRORS_AND_WARNINGS'
         }
      },
      'tenant_id': 'e721a936-7742-40b5-bf3a-eac627723ae3',
      'type': 'policy.backup.msexchange.db'
   }
```

Политика резервного копирования почтовых ящиков Microsoft Exchange Server

Политика обеспечивает функциональность резервного копирования рабочей нагрузки машины целиком.

Дополнительные сведения о параметрах планов резервного копирования см. в данном разделе пользовательской документации по продукту Кибер Бэкап Облачный.

Описание параметров запроса для создания политики резервного копирования см. данном разделе справочника API.

Следующий пример можно использовать при создании планов защиты с данной политикой:

```
{
  'enabled': true,
  'ls_features': '',
  'name': 'MS Exchange. резервная копия ящиков',
  'settings': {
    'alerts': {
      'enabled': true,
      'max_days_without_backup': 5
    },
    'archive': {
      'comments': '',
      'compression': 'MAX',
      'display_name': '',
      'format': '12',
      'name': '[Mailbox ID]_mailbox_[Plan ID]A',
      'slice_comments': '',
      'splitting': {
        'auto': true
      }
    },
    'destination': {
      'location': {
        'settings': {
          'type': 'CYBERPROTECT'
        },
        'type': 'CLOUD_STORAGE'
      },
      'performance_window': {
        'enabled': true,
        'presets': [
          {
            'cpu_priority': 'LOW',
            'id': 'HIGH',
            'speed_limit': {
              'type': 'PERCENT',
              'value': 100
            },
            'timetable': [
              {
                'days_of_week': [
                  'MON',
                  'SUN'
                ],
                'time_from': {
                  'hour': 0,
                  'minute': 0
                },
                'time_to': {
                  'hour': 23,
                  'minute': 59,
                  'second': 59
                }
```

```
        },
        {
          'days_of_week': [
            'TUE',
            'SAT'
          ],
          'time_from': {
            'hour': 0,
            'minute': 0
          },
          'time_to': {
            'hour': 18,
            'minute': 59,
            'second': 59
          }
        }
      ]
    },
    {
      'cpu_priority': 'LOW',
      'id': 'LOW',
      'speed_limit': {
        'type': 'PERCENT',
        'value': 25
      },
      'timetable': [
        {
          'days_of_week': [
            'TUE',
            'SAT'
          ],
          'time_from': {
            'hour': 19,
            'minute': 0
          },
          'time_to': {
            'hour': 23,
            'minute': 59,
            'second': 59
          }
        }
      ]
    },
    {
      'cpu_priority': 'NORMAL',
      'id': 'CANCEL',
      'speed_limit': {
        'type': 'PERCENT',
        'value': 100
      },
      'timetable': [
        {
```

```
                    'days_of_week': [
                      'WED',
                      'THU',
                      'FRI'
                    ],
                    'time_from': {
                      'hour': 0,
                      'minute': 0
                    },
                    'time_to': {
                      'hour': 23,
                      'minute': 59,
                      'second': 59
                    }
                  }
                ]
              }
            ]
          },
          'retention': {
            'execution': 'AFTER_BACKUP',
            'rules': [
              {
                'backup_set': 'DAILY',
                'max_age': {
                  'count': 7,
                  'type': 'DAYS'
                }
              },
              {
                'backup_set': 'WEEKLY',
                'max_age': {
                  'count': 4,
                  'type': 'WEEKS'
                }
              },
              {
                'backup_set': 'MONTHLY',
                'max_age': {
                  'count': 6,
                  'type': 'MONTHS'
                }
              }
            ]
          }
        },
        'error_handling': {
          'enabled': true,
          'interval': {
            'count': 30,
            'type': 'SECONDS'
          },
```

```
        'max_attempts': 10,
        'silent_mode': true
      },
      'scheduling': {
        'backup_sets': [
          {
            'schedule': {
              'alarms': {
                'time': {
                  'repeat_at': [
                    {
                      'hour': 10,
                      'minute': 15
                    }
                  ],
                  'weekdays': [
                    'MON',
                    'TUE',
                    'WED',
                    'THU',
                    'FRI'
                  ]
                }
              },
              'prevent_sleep': true,
              'type': 'WEEKLY'
            },
            'type': 'AUTO'
          }
        ],
        'enabled': true,
        'max_parallel_backups': 8,
        'rand_max_delay': {
          'count': 30,
          'type': 'MINUTES'
        },
        'scheme': 'ALWAYS_INCREMENTAL',
        'task_failure': {
          'enabled': true,
          'interval': {
            'count': 1,
            'type': 'HOURS'
          },
          'max_attempts': 1
        },
        'weekly_backup_day': 'MON'
      },
      'windows_event_log': {
        'enabled': true,
        'level': 'ERRORS_AND_WARNINGS'
      }
    },
```

```
    'tenant_id': 'e721a936-7742-40b5-bf3a-eac627723ae3',
    'type': 'policy.backup.msexchange.doc'
}
```

2.10.8.7 Создание плана резервного копирования

1. Выполните аутентификацию на платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   ```

2. Задайте переменную plan_data, а затем присвойте этой переменной объект с ключом subject, объект, содержащий ключ policy, и массив, содержащий политику резервного копирования:

   ```
   >>> plan_data = {
       'subject': {
           'policy': {
                       'enabled': true,
                       'ls_features': '',
                       'name': 'machines_plan_single_{{id_plan_name}}',
                       'settings': {
                          'alerts': {
                             'enabled': false,
                             'max_days_without_backup': 0
                          },
                          'archive': {
                             'compression': 'NORMAL',
                             'format': 'AUTO',
                             'name': '[Machine Name]-[Plan ID]-[Unique ID]A',
                             'splitting': {
                                'size': 214748364800
                             }
                          },
                          'cbt': 'DO_NOT_USE',
                          'destination': {
                             'location': {
                                'display_name': 'win10x64simple: E:\\tmp\\',
                                'settings': {
                                   'credentials': '',
                                   'name': 'win10x64simple: E:\\tmp\\',
                                   'storage_type': 'LOCAL_FOLDER',
                                   'uri': 'E:/tmp/',
                                   'use_policy_credentials': true
                                },
                             'performance_window': {
                                'enabled': false
   ```

```
    },
    'retention': {
      'execution': 'AFTER_BACKUP',
      'rules': [
        {
          'backup_set': 'DAILY',
          'max_age': {
            'count': 7,
            'type': 'DAYS'
          }
        },
        {
          'backup_set': 'WEEKLY',
          'max_age': {
            'count': 4,
            'type': 'HOURS'
          }
        }
      ]
    }
  },
  'error_handling': {
    'enabled': true,
    'interval': {
      'count': 30,
      'type': 'SECONDS'
    },
    'max_attempts': 30
  },
  'file_filters': {
    'exclude_hidden': true
  },
  'file_level_backup_snapshot': 'CREATE_WHEN_POSSIBLE',
  'fixed_drive_behavior_for_removable': true,
  'multi_volume_snapshotting_enabled': true,
  'notarization': {
    'enabled': false
  },
  'sector_by_sector':true,
  'preserve_file_security_settings': true,
  'scheduling': {
    'backup_sets': [
      {
        'schedule': {
          'alarms': {
            'time': {
              'rand_max_delay': {
                'count': 30,
                'type': 'MINUTES'
              },
              'repeat_at': [
```

```
                                {
                                    'hour': 15,
                                    'minute': 0,
                                    'second': null
                                }
                            ],
                            'weekdays': [
                                'MON',
                                'TUE',
                                'WED',
                                'THU',
                                'FRI'
                            ]
                        }
                    },
                    'prevent_sleep': true,
                    'type': 'DAILY'
                },
                'type': 'AUTO'
            }
        ],
        'enabled': true,
        'rand_max_delay': {
            'count': 30,
            'type': 'MINUTES'
        },
        'scheme': 'WEEKLY_FULL_DAILY_INCREMENTAL'
    },
    'silent_mode_enabled': true,
    'tapes': {
        'eject_media_if_successfull': false,
        'multiplexing': {
            'enabled': false,
            'streams_per_drive': 0
        },
        'multistreaming': {
            'enabled': false
        },
        'overwrite_data_on_tape': false,
        'preserve_tapes_position': true,
        'tapeset': ''
    },
    'vss': {
        'enabled': true,
        'provider': 'TARGET_SYSTEM_DEFINED'
    }
},
'tenant_id': '00000000-0000-0000-0000-000000000000',
'type': 'policy.backup.machine'
}
}
```

3. Преобразуйте объект plan_data в текст в формате JSON.

```
>>> plan_data = json.dumps(plan_data, indent=4)
```

4. Отправьте запрос POST с текстом в формате JSON на конечную точку /policies:

```
>>> response = requests.post(
...     f'{base_url}/policies',
...     headers={'Content-Type': 'application/json', **auth},
...     data=plan_data,
... )
```

5. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что план защиты создан.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, тело ответа содержит объект с ключом result в формате JSON, содержащий идентификатор политики. После преобразования в объект он будет выглядеть так:

```
>>> pprint.pprint(response.json())
{'result': ['256f2ee5-c9fd-4aad-8607-97a977deeaae@policy.backup.machine']}
```

2.10.8.8 Получение перечня планов резервного копирования

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
```

2. Отправьте запрос GET на конечную точку /policies:

```
>>> response = requests.get(f'{base_url}/policies', headers=auth)
```

3. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, тело ответа содержит объект с ключом items в формате JSON, содержащий массив объектов политик защиты. После преобразования в объект он будет выглядеть так:

```
>>> pprint.pprint(response.json())
{'items': [
      {
        'policy': {
          'created_at': '2025-10-07T21:01:10.1136754Z',
          'days_without_backup_alert': null,
          'deleted_at': null,
          'enabled': true,
          'id': 'f5e556c4-27e0-4767-b596-e48fe8d005af',
          'ls_features':
'BPCompression|BPRetentionRules|BPValidation|ForensicMode|Notary|diskSource',
          'name': ' ',
          'plan_hash': '00000000-0000-0000-0000-000000000000',
          'settings': {
            'alerts': {
              'enabled': false,
              'max_days_without_backup': 0
            },
            'archive': {
              'compression': 'MAX',
              'encryption': {
                'algorithm': 'AES192',
                'password': 'string'
              },
              'format': '12',
              'name': '[Machine Name]-[Plan ID]-[Unique ID]A[Plan ID][Plan name][Unique ID]
[Virtualization Server Type]F',
              'splitting': {
                'size': 214748364800
              }
            },
            'cbt': 'USE_IF_ENABLED',
            'check_file_timestamp': true,
            'destination': {
              'location': {
                'settings': {
                  'name': 'customer-dg09',
                  'storage_type': '',
                  'uri': 'avfs:/online?account%3dcustomer-dg09%26provider%3dCyberProtect',
                  'use_policy_credentials': true
                },
                'type': 'VAULT'
              },
              'performance_window': {
                'enabled': false
              },
              'retention': {
                'execution': 'AFTER_BACKUP',
                'rules': [
```

```
            {
               'max_count': 7
            }
         ]
      }
   },
   'error_handling': {
      'enabled': true,
      'interval': {
         'count': 1,
         'type': 'MINUTES'
      },
      'max_attempts': 1
   },
   'exclude_asz': true,
   'exclude_page_files': true,
   'fast_backup_enabled': true,
   'file_filters': {
      'exclude_hidden': true
   },
   'fixed_drive_behavior_for_removable': true,
   'forensic': {
      'dump_running_processes': false,
      'dump_type': 'RAW_MEMORY_DUMP',
      'enabled': true
   },
   'ignore_bad_sectors': true,
   'lvm_snapshotting_enabled': true,
   'multi_volume_snapshotting_enabled': true,
   'notarization': {
      'enabled': true
   },
   'owner_id': '32e95e91-39f4-6053-9ce4-0da7e86c8b81',
   'preserve_file_security_settings': true,
   'reset_free_blocks_list': true,
   'scheduling': {
      'backup_sets': [
         {
            'schedule': {
               'alarms': {
                  'time': {
                     'rand_max_delay': {
                        'count': 30,
                        'type': 'MINUTES'
                     },
                     'repeat_at': [
                        {
                           'hour': 9,
                           'minute': 0
                        }
                     ],
```

```
                        'weekdays': [
                          'MON',
                          'TUE',
                          'WED'
                        ]
                      }
                    },
                    'conditions': {},
                    'prevent_sleep': true,
                    'type': 'DAILY'
                  },
                  'type': 'AUTO'
                }
              ],
              'enabled': true,
              'rand_max_delay': {
                'count': 30,
                'type': 'MINUTES'
              },
              'scheme': 'WEEKLY_FULL_DAILY_INCREMENTAL',
              'weekly_backup_day': 'MON'
            },
            'sector_by_sector': true,
            'silent_mode_enabled': true,
            'tapes': {
              'devices': null,
              'eject_media_if_successfull': false,
              'multiplexing': {
                'enabled': false,
                'streams_per_drive': 0
              },
              'multistreaming': {
                'enabled': false
              },
              'overwrite_data_on_tape': false,
              'preserve_tapes_position': true,
              'tapeset': ''
            },
            'validate_backup': true,
            'vss': {
              'enabled': true,
              'provider': 'NATIVE'
            }
          },
          'source_type': 'machines',
          'tenant_id': '00000000-0000-0000-0000-000000000000',
          'type': 'policy.backup.machine',
          'updated_at': '2025-10-07T21:01:10.1131115Z'
        }
      },
      {
```

```
        'policy': {
          'created_at': '2025-10-05T00:58:55.1236553Z',
          'days_without_backup_alert': null,
          'deleted_at': null,
          'enabled': true,
          'id': 'cffe3239-102c-40f0-ac9a-f0f5520b3f9e',
          'ls_features':
'BPCompression|BPRetentionRules|BPValidation|ForensicMode|Notary|diskSource',
          'name': '89',
          'plan_hash': '00000000-0000-0000-0000-000000000000',
          'settings': {
            'alerts': {
              'enabled': false,
              'max_days_without_backup': 0
            },
            'archive': {
              'compression': 'MAX',
              'encryption': {
                'algorithm': 'AES192',
                'password': 'string'
              },
              'format': '12',
              'name': '[Machine Name]-[Plan ID]-[Unique ID]A[Plan ID][Plan name][Unique ID]
[Virtualization Server Type]F',
              'splitting': {
                'size': 214748364800
              }
            },
            'cbt': 'USE_IF_ENABLED',
            'check_file_timestamp': true,
            'destination': {
              'location': {
                'settings': {
                  'name': 'customer-dg09',
                  'storage_type': '',
                  'uri': 'avfs:/online?account%3dcustomer-dg09%26provider%3dCyberProtect',
                  'use_policy_credentials': true
                },
                'type': 'VAULT'
              },
              'performance_window': {
                'enabled': false
              },
              'retention': {
                'execution': 'AFTER_BACKUP',
                'rules': [
                  {
                    'max_count': 7
                  }
                ]
              }
```

```
},
'error_handling': {
   'enabled': true,
   'interval': {
      'count': 1,
      'type': 'MINUTES'
   },
   'max_attempts': 1
},
'exclude_asz': true,
'exclude_page_files': true,
'fast_backup_enabled': true,
'file_filters': {
   'exclude_hidden': true
},
'fixed_drive_behavior_for_removable': true,
'forensic': {
   'dump_running_processes': false,
   'dump_type': 'RAW_MEMORY_DUMP',
   'enabled': true
},
'ignore_bad_sectors': true,
'lvm_snapshotting_enabled': true,
'multi_volume_snapshotting_enabled': true,
'notarization': {
   'enabled': true
},
'owner_id': '32e95e91-39f4-6053-9ce4-0da7e86c8b81',
'preserve_file_security_settings': true,
'reset_free_blocks_list': true,
'scheduling': {
   'backup_sets': [
      {
         'schedule': {
            'alarms': {
               'time': {
                  'rand_max_delay': {
                     'count': 30,
                     'type': 'MINUTES'
                  },
                  'repeat_at': [
                     {
                        'hour': 9,
                        'minute': 0
                     }
                  ],
                  'weekdays': [
                     'MON',
                     'TUE',
                     'WED'
                  ]
```

```
                        }
                    },
                    'conditions': {},
                    'prevent_sleep': true,
                    'type': 'DAILY'
                },
                'type': 'AUTO'
            }
        ],
        'enabled': true,
        'rand_max_delay': {
            'count': 30,
            'type': 'MINUTES'
        },
        'scheme': 'WEEKLY_FULL_DAILY_INCREMENTAL',
        'weekly_backup_day': 'MON'
    },
    'sector_by_sector': true,
    'silent_mode_enabled': true,
    'tapes': {
        'devices': null,
        'eject_media_if_successfull': false,
        'multiplexing': {
            'enabled': false,
            'streams_per_drive': 0
        },
        'multistreaming': {
            'enabled': false
        },
        'overwrite_data_on_tape': false,
        'preserve_tapes_position': true,
        'tapeset': ''
    },
    'validate_backup': true,
    'vss': {
        'enabled': true,
        'provider': 'NATIVE'
    }
},
'source_type': 'machines',
'tenant_id': '00000000-0000-0000-0000-000000000000',
'type': 'policy.backup.machine',
'updated_at': '2025-10-05T00:58:55.1217739Z'
}
},
{
'policy': {
    'created_at': '2025-10-04T13:32:54.5440459Z',
    'days_without_backup_alert': null,
    'deleted_at': null,
    'enabled': true,
```

```
            'id': '0183b910-0107-4ea8-b6b0-dedd2e2f5f8d',
            'ls_features': 'BPCompression|BPRetentionRules|BPValidation|Notary|fileSource',
            'name': 'rename_12',
            'plan_hash': 'B84AE99C-05BA-39BF-9E40-36AE4FF5FE51',
            'settings': {
              'alerts': {
                'enabled': false,
                'max_days_without_backup': 0
              },
              'archive': {
                'compression': 'MAX',
                'encryption': {
                  'algorithm': 'AES192',
                  'password': 'string'
                },
                'format': '12',
                'name': '[Machine Name]-[Plan ID]-[Unique ID]A[Plan ID][Plan name][Unique ID]
[Virtualization Server Type]F',
                'splitting': {
                  'size': 214748364800
                }
              },
              'destination': {
                'location': {
                  'settings': {
                    'name': 'customer-dg09',
                    'storage_type': '',
                    'uri': 'avfs:/online?account%3dcustomer-dg09%26provider%3dCyberProtect',
                    'use_policy_credentials': true
                  },
                  'type': 'VAULT'
                },
                'performance_window': {
                  'enabled': false
                },
                'retention': {
                  'execution': 'AFTER_BACKUP',
                  'rules': [
                    {
                      'max_count': 7
                    }
                  ]
                }
              },
              'error_handling': {
                'enabled': true,
                'interval': {
                  'count': 1,
                  'type': 'MINUTES'
                },
                'max_attempts': 1
```

```
},
'file_filters': {
   'exclude_hidden': true
},
'file_level_backup_snapshot': 'CREATE_WHEN_POSSIBLE',
'fixed_drive_behavior_for_removable': true,
'ignore_bad_sectors': true,
'multi_volume_snapshotting_enabled': true,
'notarization': {
   'enabled': true
},
'owner_id': '32e95e91-39f4-6053-9ce4-0da7e86c8b81',
'preserve_file_security_settings': true,
'scheduling': {
   'backup_sets': [
      {
         'schedule': {
            'alarms': {
               'time': {
                  'rand_max_delay': {
                     'count': 30,
                     'type': 'MINUTES'
                  },
                  'repeat_at': [
                     {
                        'hour': 9,
                        'minute': 0
                     }
                  ],
                  'weekdays': [
                     'MON',
                     'TUE',
                     'WED'
                  ]
               }
            },
            'conditions': {},
            'prevent_sleep': true,
            'type': 'DAILY'
         },
         'type': 'AUTO'
      }
   ],
   'enabled': true,
   'rand_max_delay': {
      'count': 30,
      'type': 'MINUTES'
   },
   'scheme': 'WEEKLY_FULL_DAILY_INCREMENTAL',
   'weekly_backup_day': 'MON'
},
```

```
            'selection': {},
            'silent_mode_enabled': true,
            'tapes': {
              'devices': null,
              'eject_media_if_successfull': false,
              'multiplexing': {
                'enabled': false,
                'streams_per_drive': 0
              },
              'multistreaming': {
                'enabled': false
              },
              'overwrite_data_on_tape': false,
              'preserve_tapes_position': true,
              'tapeset': ''
            },
            'validate_backup': true,
            'vss': {
              'enabled': true,
              'provider': 'NATIVE'
            }
          },
          'source_type': 'files',
          'tenant_id': '00000000-0000-0000-0000-000000000000',
          'type': 'policy.backup.files',
          'updated_at': '2025-10-04T23:11:21.3135901Z'
        }
      }],
 'paging': {'cursors': {'after': ''}}}
```


4. Преобразуйте текст в формате JSON, содержащийся в теле ответа, в объект, а затем извлеките список политик из ответа:

```
>>> policies = response.json()['items']
```

2.10.8.9 Получение информации о плане резервного копирования

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
```

2. Присвойте переменной policy_id UUID политики, созданной с помощью API (см. "Создание плана резервного копирования" (стр. 125)), или политики, найденной с помощью поиска (см. "Получение перечня планов резервного копирования" (стр. 128)):

```
>>> policy_id = created_policy_id
>>> policy_id
'dac6c7c4-6b3f-4b46-9aa8-b78a189a0f10'
```

3. Отправьте запрос GET на конечную точку /policies/<policy_id>. Для отображения параметров раздела settings URL-адрес конечной точки должен содержать параметр запроса include_settings=true, например:

```
>>> params = {'include_settings': true}
>>> response = requests.get(f'{base_url}/policies/{policy_id}', headers=auth, params=params)
```

4. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, тело ответа содержит объект с ключом items в формате JSON, содержащий массив с одним объектом политики защиты. После преобразования в объект он будет выглядеть так:

```
>>> pprint.pprint(response.json())
{'items': [
      {
        'policy': {
          'created_at': '2025-10-07T21:01:10.1136754Z',
          'days_without_backup_alert': null,
          'deleted_at': null,
          'enabled': true,
          'id': 'f5e556c4-27e0-4767-b596-e48fe8d005af',
          'ls_features':
'BPCompression|BPRetentionRules|BPValidation|ForensicMode|Notary|diskSource',
          'name': ' ',
          'plan_hash': '00000000-0000-0000-0000-000000000000',
          'settings': {
            'alerts': {
              'enabled': false,
              'max_days_without_backup': 0
            },
            'archive': {
              'compression': 'MAX',
              'encryption': {
                'algorithm': 'AES192',
                'password': 'string'
              },
              'format': '12',
              'name': '[Machine Name]-[Plan ID]-[Unique ID]A[Plan ID][Plan name][Unique ID]
[Virtualization Server Type]F',
```

```
    'splitting': {
      'size': 214748364800
    }
  },
  'cbt': 'USE_IF_ENABLED',
  'check_file_timestamp': true,
  'destination': {
    'location': {
      'settings': {
        'name': 'customer-dg09',
        'storage_type': '',
        'uri': 'avfs:/online?account%3dcustomer-dg09%26provider%3dCyberProtect',
        'use_policy_credentials': true
      },
      'type': 'VAULT'
    },
    'performance_window': {
      'enabled': false
    },
    'retention': {
      'execution': 'AFTER_BACKUP',
      'rules': [
        {
          'max_count': 7
        }
      ]
    }
  },
  'error_handling': {
    'enabled': true,
    'interval': {
      'count': 1,
      'type': 'MINUTES'
    },
    'max_attempts': 1
  },
  'exclude_asz': true,
  'exclude_page_files': true,
  'fast_backup_enabled': true,
  'file_filters': {
    'exclude_hidden': true
  },
  'fixed_drive_behavior_for_removable': true,
  'forensic': {
    'dump_running_processes': false,
    'dump_type': 'RAW_MEMORY_DUMP',
    'enabled': true
  },
  'ignore_bad_sectors': true,
  'lvm_snapshotting_enabled': true,
  'multi_volume_snapshotting_enabled': true,
```

```
'notarization': {
   'enabled': true
},
'owner_id': '32e95e91-39f4-6053-9ce4-0da7e86c8b81',
'preserve_file_security_settings': true,
'reset_free_blocks_list': true,
'scheduling': {
   'backup_sets': [
      {
         'schedule': {
            'alarms': {
               'time': {
                  'rand_max_delay': {
                     'count': 30,
                     'type': 'MINUTES'
                  },
                  'repeat_at': [
                     {
                        'hour': 9,
                        'minute': 0
                     }
                  ],
                  'weekdays': [
                     'MON',
                     'TUE',
                     'WED'
                  ]
               }
            },
            'conditions': {},
            'prevent_sleep': true,
            'type': 'DAILY'
         },
         'type': 'AUTO'
      }
   ],
   'enabled': true,
   'rand_max_delay': {
      'count': 30,
      'type': 'MINUTES'
   },
   'scheme': 'WEEKLY_FULL_DAILY_INCREMENTAL',
   'weekly_backup_day': 'MON'
},
'sector_by_sector': true,
'silent_mode_enabled': true,
'tapes': {
   'devices': null,
   'eject_media_if_successfull': false,
   'multiplexing': {
      'enabled': false,
```

```
                    'streams_per_drive': 0
                  },
                  'multistreaming': {
                    'enabled': false
                  },
                  'overwrite_data_on_tape': false,
                  'preserve_tapes_position': true,
                  'tapeset': ''
                },
                'validate_backup': true,
                'vss': {
                  'enabled': true,
                  'provider': 'NATIVE'
                }
              },
              'source_type': 'machines',
              'tenant_id': '00000000-0000-0000-0000-000000000000',
              'type': 'policy.backup.machine',
              'updated_at': '2025-10-07T21:01:10.1131115Z'
            }
          }
 ],
  'paging': {'cursors': {'after': ''}}}
```


5. Преобразуйте текст в формате JSON, содержащийся в теле ответа, в объект, а затем извлеките данные о политике из ответа:

```
>>> policies = response.json()['items']
```

2.10.8.10 Редактирование параметров плана резервного копирования

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
```

2. Запросите политики, как описано в разделе "Получение перечня планов резервного копирования" (стр. 128). Задайте переменную policy_id и присвойте ей в качестве значения идентификатор политики. В данном примере использован идентификатор первой политики:

```
>>> policy_id = policies[0]['id']
>>> policy_id
33965f81-7293-45d4-9f13-dc4281d7bdfd
```

3. Задайте переменную policy_update, а затем присвойте этой переменной объект с ключом subject, содержащий параметры:

```
>>> policy_update = {
...     "subject": {
...         "policy": [
...             {
...                 "settings": {
...                  "alerts": {
...                     "enabled": true,
...                     "max_days_without_backup": 5 }
...                 }
...             }
...         ]
...     }
... }
```

Примечание

Список доступных параметров политики приведен в справочнике API.

4. Преобразуйте объект policy_update в текст в формате JSON.

```
>>> policy_update = json.dumps(policy_update, indent=4)
```

5. Отправьте запрос PATCH с текстом в формате JSON на конечную точку /policies/{policy_id}:

```
>>> response = requests.patch(
...     f'{base_url}/policies/{policy_id}',
...     headers={'Content-Type': 'application/json', **auth},
...     data=policy_update,
... )
```

6. Проверьте код состояния запроса:

```
>>> response.status_code
204
```

Код состояния HTTP 204 означает, что план был успешно обновлен.

Код состояния HTTP 200 означает, что план был обновлен с предупреждениями.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

2.10.8.11 Включение/выключение плана резервного копирования

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
```

2. Запросите политики, как описано в разделе "Получение перечня планов резервного копирования" (стр. 128). Задайте переменную policy_id и присвойте ей в качестве значения идентификатор политики. В данном примере использован идентификатор первой политики:

```
>>> policy_id = policies[0]['id']
>>> policy_id
33965f81-7293-45d4-9f13-dc4281d7bdfd
```

3. Задайте переменную policy_enabled, а затем присвойте этой переменной объект:

```
>>> policy_enabled = {
...   "policy_ids": [{policy_id}],
...   "value": true
... }
```

Примечание

Для выключения плана требуется передать "value": false.

Для включения или выключения плана на определённых устройствах требуется передать "context_ids": [{context_id}, ...].

4. Преобразуйте объект policy_enabled в текст в формате JSON.

```
>>> policy_enabled = json.dumps(policy_enabled, indent=4)
```

5. Отправьте запрос PATCH с текстом в формате JSON на конечную точку /policy_applications/enabled:

```
>>> response = requests.patch(
...     f'{base_url}/policy_applications/enabled',
...     headers={'Content-Type': 'application/json', **auth},
...     data=policy_enabled,
... )
```

6. Проверьте код состояния запроса:

```
>>> response.status_code
204
```

Код состояния HTTP 204 означает, что план был успешно обновлен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

2.10.8.12 Удаление плана резервного копирования

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   ```

2. Запросите планы защиты, как описано в разделе "Получение перечня планов резервного копирования" (стр. 128). Задайте переменную policy_id и присвойте ей в качестве значения идентификатор плана защиты. В данном примере использован идентификатор первого плана защиты:

   ```
   >>> policy_id = policies[0]['id']
   >>> policy_id
   5b15f6e1-88ec-4dce-b523-0e8394c0bc19
   ```

3. Отправьте запрос DELETE на конечную точку /policies/{policy_id}:

   ```
   >>> response = requests.delete(f'{base_url}/policies/{policy_id}', headers=auth)
   ```

4. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   204
   ```

   Код состояния HTTP 204 означает, что план защиты был успешно удален.

   Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

2.10.9 Применение политики

Применение политики (плана резервного копирования) предоставляет информацию о следующих статусах:

- статус развёртывания политики;
- статус выполнения политики;
- статус процесса применения политики.

Все операции с применением политик находятся в конечной точке /policy_applications.

2.10.9.1 Структура объекта JSON применения политики

Описание структуры объекта JSON применения политики доступно по ссылке.

2.10.9.2 Справочники

Справочники доступны по ссылке.

2.10.9.3 Действия с применениями политик

Операция	Используемые методы и конечные точки
"Получение списка применений плана резервного копирования" (стр. 145)	GET /policy_applications
"Применение ранее созданного плана резервного копирования к ресурсу" (стр. 147)	POST /policy_applications
"Отзыв ранее созданного плана резервного копирования с ресурса" (стр. 148)	DELETE /policy_applications
"Запуск плана резервного копирования" (стр. 150)	POST /policy_applications:run

2.10.9.4 Получение списка применений плана резервного копирования

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   ```

2. Запросите ресурсы, как описано в разделе "Получение списка ресурсов" (стр. 96). Задайте переменную resource_id и присвойте ей в качестве значения идентификатор ресурса. В данном примере использован идентификатор первого ресурса:

   ```
   >>> resource_id = resources[0]['id']
   >>> resource_id
   'b19cf0fe-95eb-4657-a743-9d4ce6bc34f6'
   ```

3. Задайте переменную filters, а затем присвойте объекту, содержащему идентификатор ресурса в ключе context_id, эту переменную:

   ```
   >>> filters = {
   ...     'context_id': resource_id
   ... }
   ```

Примечание

Чтобы получить доступ к планам, применимым к нескольким ресурсам, укажите идентификаторы ресурсов в следующем формате: or(5b15f6e1-88ec-4dce-b523-0e8394c0bc19,c70134c4-a244-4b22-99ad-e081301f7530).

4. Отправьте запрос GET на конечную точку /policy_applications:

```
>>> response = requests.get(f'{base_url}/policy_applications', headers=auth, params=filters)
```

5. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, тело ответа содержит объект с ключом items в формате JSON, содержащий массив объектов политик защиты. После преобразования в объект он будет выглядеть так:

```
>>> pprint.pprint(response.json())
{
   'items': [
      [
         {
            'agent_id': '0d47f411-3b67-42c8-b9d5-33a48b2bbe82',
            'context': {
               'id': 'b19cf0fe-95eb-4657-a743-9d4ce6bc34f6',
               'type': 'resource.machine'
            },
            'created_at': '2025-10-29T11:13:51.1782663Z',
            'deleted_at': null,
            'deployment': {
               'state': 'DEPLOYED'
            },
            'enabled': true,
            'id': '7d61b4b9-8f39-42ad-824c-d508a6ccbf59',
            'last_activity': 'policy.backup.machine',
            'next_activity': 'policy.backup.machine',
            'next_run_time': '2025-10-29T20:00:00Z',
            'origin_contexts': [
               'b19cf0fe-95eb-4657-a743-9d4ce6bc34f6'
            ],
            'policy': {
               'id': '5ea704d9-a214-4e81-97d9-d228098f8431',
               'name': 'Новый план защиты',
               'type': 'policy.backup.machine'
            },
            'running': {
```

```
                'state': 'IDLE'
            },
            'status': 'OK',
            'tenant_id': '00000000-0000-0000-0000-000000000000',
            'updated_at': '2025-10-29T11:13:51.7559748Z'
        }
    ]
  ],
  'paging': {
    'cursors': {
      'total': 1
    }
  }
}
```


2.10.9.5 Применение ранее созданного плана резервного копирования к ресурсу

1. Выполните аутентификацию на платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   ```

2. Запросите планы резервного копирования, как описано в разделе "Получение перечня планов резервного копирования" (стр. 128). Задайте переменную policy_id и присвойте ей в качестве значения идентификатор плана резервного копирования. В данном примере использован идентификатор первого плана резервного копирования:

   ```
   >>> policy_id = policies[0]['id']
   >>> policy_id
   5b15f6e1-88ec-4dce-b523-0e8394c0bc19
   ```

3. Запросите ресурсы, как описано в разделе "Получение списка ресурсов" (стр. 96). Задайте переменную resource_id и присвойте ей в качестве значения идентификатор ресурса. В данном примере использован идентификатор первого ресурса:

   ```
   >>> resource_id = resources[0]['id']
   >>> resource_id
   '5c350066-2ba6-4eeb-aa91-1213dd35f033'
   ```

4. Задайте переменную application_data, а затем назначьте объекту, содержащему ключ policy_id, ID плана резервного копирования, а объекту context — ключ items, содержащий список идентификаторов ресурсов, для этой переменной:

```
>>> application_data = {
...     'policy_id': policy_id,
...     'context': {
...         'items': [
...             resource_id
...         ]
...     }
... }
```

5. Преобразуйте объект application_data в текст в формате JSON.

```
>>> application_data = json.dumps(application_data, indent=4)
```

6. Отправьте запрос POST с текстом в формате JSON на конечную точку /policy_applications:

```
>>> response = requests.post(
...     f'{base_url}/policy_applications',
...     headers={'Content-Type': 'application/json', **auth},
...     data=application_data,
... )
```

7. Проверьте код состояния запроса:

```
>>> response.status_code
204
```

Код состояния HTTP 204 означает, что план резервного копирования был применен к выбранным ресурсам.

Код состояния HTTP 200 означает, что план резервного копирования был применен к выбранным ресурсам с некритическими предупреждениями.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Тело ответа с кодом состояния HTTP 200 содержит объект с ключом issues в формате JSON, содержащий массив с предупреждениями. После преобразования в объект он будет выглядеть так:

```
>>> pprint.pprint(response.json())
{'issues': [...]}
```

2.10.9.6 Отзыв ранее созданного плана резервного копирования с ресурса

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
```

```
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
```

2. Запросите планы резервного копирования, как описано в разделе "Получение перечня планов резервного копирования" (стр. 128). Задайте переменную policy_id и присвойте ей в качестве значения идентификатор плана резервного копирования. В данном примере использован идентификатор первого плана резервного копирования:

```
>>> policy_id = policies[0]['id']
>>> policy_id
5b15f6e1-88ec-4dce-b523-0e8394c0bc19
```

3. Запросите ресурсы, как описано в разделе "Получение списка ресурсов" (стр. 96). Задайте переменную resource_id и присвойте ей в качестве значения идентификатор ресурса. В данном примере использован идентификатор первого ресурса:

```
>>> resource_id = resources[0]['id']
>>> resource_id
'5c350066-2ba6-4eeb-aa91-1213dd35f033'
```

4. Задайте переменную filters, а затем присвойте этой переменной объект с ключом policy_id, содержащим ID политики, и ключом context, содержащим ID ресурса:

```
>>> filters = {
...     'policy_id': policy_id,
...     'context_id': resource_id
... }
```

Примечание

Чтобы отозвать планы из нескольких ресурсов, укажите значения в следующем формате: or (33965f81-7293-45d4-9f13-dc4281d7bdfd,c70134c4-a244-4b22-99ad-e081301f7530).

Список доступных параметров запроса приведен в справочнике API.

5. Отправьте запрос DELETE на конечную точку /policy_applications:

```
>>> response = requests.delete(
...     f'{base_url}/policy_applications',
...     headers=auth,
...     params=filters,
... )
```

6. Проверьте код состояния запроса:

```
>>> response.status_code
204
```

Код состояния HTTP 204 означает, что план резервного копирования был успешно отозван с выбранных ресурсов.

Код состояния HTTP 200 означает, что план резервного копирования был отозван с выбранных ресурсов с некритическими предупреждениями.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Тело ответа с кодом состояния HTTP 200 содержит объект с ключом issues в формате JSON, содержащий массив с предупреждениями. После преобразования в объект он будет выглядеть так:

```
>>> pprint.pprint(response.json())
{'issues': [...]}
```

2.10.9.7 Запуск плана резервного копирования

1. Выполните аутентификацию на платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   ```

2. Запросите планы резервного копирования, как описано в разделе "Получение перечня планов резервного копирования" (стр. 128). Задайте переменную policy_id и присвойте ей в качестве значения идентификатор плана резервного копирования. В данном примере использован идентификатор первого плана резервного копирования:

   ```
   >>> policy_id = policies[0]['id']
   >>> policy_id
   33965f81-7293-45d4-9f13-dc4281d7bdfd
   ```

3. [Необязательно] Запросите ресурсы, как описано в разделе "Получение списка ресурсов" (стр. 96). Задайте переменную resource_id и присвойте ей в качестве значения идентификатор ресурса. В данном примере использован идентификатор первого ресурса:

   ```
   >>> resource_id = resources[0]['id']
   >>> resource_id
   '5c350066-2ba6-4eeb-aa91-1213dd35f033'
   ```

   Примечание

   Если в запросе не указаны ресурсы, политика будет выполнена для всех ресурсов, к которым она применяется.

4. Задайте переменную policy_exec_data, а затем присвойте этой переменной объект, содержащий следующие параметры JSON:

```
>>> policy_exec_data = {
...     'state': 'RUNNING',
...     'policy_id': policy_id,
...     'context_ids': [resource_id]
... }
```

Примечание

При передаче 'state': 'IDLE' будет выполнена остановка плана резервного копирования.

Список доступных параметров запроса приведен в справочнике API.

5. Преобразуйте объект policy_exec_data в текст в формате JSON:

```
>>> policy_exec_data = json.dumps(policy_exec_data, indent=4)
```

6. Отправьте запрос POST с текстом в формате JSON на конечную точку /policy_applications:run:

```
>>> response = requests.post(
...     f'{base_url}/policy_applications:run',
...     headers={'Content-Type': 'application/json', **auth},
...     data=policy_exec_data,
... )
```

7. Проверьте код состояния запроса:

```
>>> response.status_code
202
```

Код состояния HTTP 202 означает, что запрос был принят на выполнение.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

2.10.10 Реквизиты для входа

Реквизиты для входа – это учётные данные (имена учетных записей для входа, пароли, ключи доступа) для обеспечения доступа. Применяются для выполнения операций над объектами, требующими авторизации при доступе.

2.10.10.1 Типы реквизитов для входа

Типы	Подтипы
account	Учётные записи для авторизации. Используемые подтипы: • computer • esx • vcenter • scale • ovirt

Типы	Подтипы
	• ms_exchange • vm_guest_system • ldap
archive	Учётные данные архивов. Используемые подтипы: • endpoint • ''
share	Учётные данные сетевых папок. Используемые подтипы: • smb • nfs • sftp
asz	Учётные данные Зоны безопасности. Подтипы отсутствуют.
db	Учётные данные баз данных. Используемые подтипы: • mysql • mssql
folder	Учётные данные Локальных папок. Подтипы отсутствуют.

Все операции с реквизитами для входа находятся в конечной точке /credentials.

2.10.10.2 Структура объекта JSON реквизитов для входа

Описание структуры объекта JSON реквизитов для входа доступно по ссылке.

2.10.10.3 Справочники

Справочники доступны по ссылке.

2.10.10.4 Действия с реквизитами для входа

Операция	Используемые методы и конечные точки
"Создание реквизитов для входа" (стр. 153)	POST /credentials
"Получение информации о реквизитах для входа" (стр. 154)	GET /credentials/{credentials_id}
"Удаление реквизитов для входа" (стр. 155)	DELETE /credentials/{credentials_id}

2.10.10.5 Создание реквизитов для входа

1. Выполните аутентификацию на платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   ```

2. Задайте переменную cred_data, а затем присвойте этой переменной объект:

   ```
   >>> cred_data = {
      'scope': {
         'type': 'folder'
         },
      'description': 'новое описание',
      'identity': 'user_name',
      'secret': 'user_pass'
      }
   ```

3. Преобразуйте объект cred_data в текст в формате JSON.

   ```
   >>> cred_data = json.dumps(cred_data, indent=4)
   ```

4. Отправьте запрос POST с текстом в формате JSON на конечную точку /credentials:

   ```
   >>> response = requests.post(
   ...    f'{base_url}/credentials',
   ...    headers={'Content-Type': 'application/json', **auth},
   ...    data=cred_data,
   ... )
   ```

5. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   201
   ```

   Код состояния HTTP 201 означает, что реквизиты для входа созданы.

   Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

   Кроме того, тело ответа содержит объект с ключом id в формате JSON, содержащий идентификатор реквизитов для входа. После преобразования в объект он будет выглядеть так:

   ```
   >>> pprint.pprint(response.json())
   {
      'id': 'b12e9914-ecea-4869-8e55-3e62a6dd8bb8'
   }
   ```

2.10.10.6 Получение информации о реквизитах для входа

1. Выполните аутентификацию на платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   ```

2. Присвойте переменной credentials_id UUID реквизитов для входа, созданных с помощью API (см. "Создание реквизитов для входа" (стр. 153)):

   ```
   >>> credentials_id = created_credentials_id
   >>> credentials_id
   'b12e9914-ecea-4869-8e55-3e62a6dd8bb8'
   ```

3. Отправьте запрос GET на конечную точку /credentials/<credentials_id>:

   ```
   >>> response = requests.get(f'{base_url}/credentials/{credentials_id}', headers=auth)
   ```

4. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   200
   ```

   Код состояния HTTP 200 означает, что запрос был успешно выполнен.

   Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

   Кроме того, тело ответа содержит объект в формате JSON, содержащий параметры реквизитов для входа. После преобразования в объект он будет выглядеть так:

   ```
   >>> pprint.pprint(response.json())
   {
      'scope': {
         'type': 'folder',
         'subtype': ''
      },
      'description': 'новое описание',
      'identity': 'user_name',
      'id': 'b12e9914-ecea-4869-8e55-3e62a6dd8bb8',
      'tenant': '8f2ce031-3a88-448a-b6b6-84fc8728eab4',
      'version': 0,
      'secret_autogenerated': true,
      'created_at': '2025-11-05T14:15:50Z',
      'updated_at': '2025-11-05T14:15:50Z'
   }
   ```

2.10.10.7 Удаление реквизитов для входа

1. Выполните аутентификацию на платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   ```

2. Присвойте переменной credentials_id UUID реквизитов для входа, созданных с помощью API (см. "Создание реквизитов для входа" (стр. 153)):

   ```
   >>> credentials_id = created_credentials_id
   >>> credentials_id
   'b12e9914-ecea-4869-8e55-3e62a6dd8bb8'
   ```

3. Отправьте запрос DELETE на конечную точку /credentials/{credentials_id}:

   ```
   >>> response = requests.delete(f'{base_url}/credentials/{credentials_id}', headers=auth)
   ```

4. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   204
   ```

   Код состояния HTTP 204 означает, что план защиты был успешно удален.

   Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

2.10.11 Архив резервных копий

Архив – файл, содержащий набор резервных копий (точек востановления) и метаданные для них. Файл содержит цепочку резервных копий: полную или дифференциальную резервную копию и все зависящие от неё инкрементные копии. Архив формируется при создании хотя бы одной резервной копии. В дальнейшем он используется для хранения данных при резервном копировании и восстановлении. В архиве применяется сжатие и дедупликация данных.

Все операции с архивами находятся в конечной точке /archives.

2.10.11.1 Структура объекта JSON архива

Описание структуры объекта JSON архива доступно по ссылке.

2.10.11.2 Справочники

Справочники доступны по ссылке.

2.10.11.3 Действия с архивами

Операция	Используемые методы и конечные точки
"Просмотр архивов резервных копий" (стр. 156)	GET /archives

2.10.11.4 Просмотр архивов резервных копий

1. Выполните аутентификацию на платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   ```

2. Отправьте запрос GET на конечную точку /archives:

   ```
   >>> response = requests.get(f'{base_url}/archives', headers=auth)
   ```

   Примечание
   Список доступных параметров строки запроса приведен в справочнике API.

3. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   200
   ```

   Код состояния HTTP 200 означает, что запрос был успешно выполнен.

   Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

   Кроме того, тело ответа содержит массив items в формате JSON, содержащий данные архивов. После преобразования в объект он будет выглядеть так:

   ```
   >>> pprint.pprint(response.json())
   {
     'items': [
       {
         'id': '5ab0433d-991a-3d0f-1328-4b1874d621a3',
         'name': 'AAURxWO@odio_mock_200_account_workmail_A630C04C-14F1-32BA-9585-
   2C9472EF3E00_edd0ec02-67df-4017-abd8-ee37611dbeb5A',
         'vault_id': '21882913-cc56-646e-1036-9eac88c3977a',
         'size': 20480,
         'compressed_data_size': 0,
         'data_size': 0,
   ```

```
        'original_data_size': 0,
        'logical_size': 0,
        'type': 'EMPTY',
        'format': '12',
        'created_at': '1970-01-01T00:00:00Z',
        'updated_at': '0001-01-01T00:00:00Z',
        'last_backup_created_at': '1970-01-01T00:00:00Z',
        'protected_by_password': false,
        'encryption_algorithm': 'NONE',
        'deleted': false,
        'file_name': '',
        'tenant_id': 'f9cf6b51-90c8-43bb-8da5-1d6018de265e',
        'agent_id': '00000000-0000-0000-0000-000000000000',
        'agent_name': '',
        'description': '',
        'display_name': 'AAURxWO@odio_mock_200_account_workmail_A630C04C-14F1-
32BA-9585-2C9472EF3E00_edd0ec02-67df-4017-abd8-ee37611dbeb5A',
        'owner_id': '',
        'owner_name': '',
        'resource_id': '',
        'consistent': false,
        'marked_for_deletion': false,
        'last_seen_at': '2025-09-29T15:59:09Z',
        'vault': {
          'id': '21882913-cc56-646e-1036-9eac88c3977a',
          'name': 'pk-customer',
          'storage_type': 'ONLINE'
        }
      },
      {
        'id': 'b9ac73cc-e091-d511-2205-965b46a24e71',
        'name': 'AbouXIf@odio_mock_200_account_workmail_C9DABFE4-02E7-34B1-8FF0-
239E0E69DD84_edd0ec02-67df-4017-abd8-ee37611dbeb5A',
        'vault_id': '21882913-cc56-646e-1036-9eac88c3977a',
        'size': 20480,
        'compressed_data_size': 0,
        'data_size': 0,
        'original_data_size': 0,
        'logical_size': 0,
        'type': 'EMPTY',
        'format': '12',
        'created_at': '1970-01-01T00:00:00Z',
        'updated_at': '0001-01-01T00:00:00Z',
        'last_backup_created_at': '1970-01-01T00:00:00Z',
        'protected_by_password': false,
        'encryption_algorithm': 'NONE',
        'deleted': false,
        'file_name': '',
        'tenant_id': 'f9cf6b51-90c8-43bb-8da5-1d6018de265e',
        'agent_id': '00000000-0000-0000-0000-000000000000',
        'agent_name': '',
```

```
        'description': '',
        'display_name': 'AbouXIf@odio_mock_200_account_workmail_C9DABFE4-02E7-34B1-
8FF0-239E0E69DD84_edd0ec02-67df-4017-abd8-ee37611dbeb5A',
        'owner_id': '',
        'owner_name': '',
        'resource_id': '',
        'consistent': false,
        'marked_for_deletion': false,
        'last_seen_at': '2025-09-29T15:59:09Z',
        'vault': {
          'id': '21882913-cc56-646e-1036-9eac88c3977a',
          'name': 'pk-customer',
          'storage_type': 'ONLINE'
        }
      },
      {
        'id': 'c60019f2-98ce-b1ec-af1a-0db0fb2644e2',
        'name': 'aajsSxM@odio_mock_200_account_workmail_67028231-BFC9-352E-A3D9-
0D934164DDC3_edd0ec02-67df-4017-abd8-ee37611dbeb5A',
        'vault_id': '21882913-cc56-646e-1036-9eac88c3977a',
        'size': 20480,
        'compressed_data_size': 0,
        'data_size': 0,
        'original_data_size': 0,
        'logical_size': 0,
        'type': 'EMPTY',
        'format': '12',
        'created_at': '1970-01-01T00:00:00Z',
        'updated_at': '0001-01-01T00:00:00Z',
        'last_backup_created_at': '1970-01-01T00:00:00Z',
        'protected_by_password': false,
        'encryption_algorithm': 'NONE',
        'deleted': false,
        'file_name': '',
        'tenant_id': 'f9cf6b51-90c8-43bb-8da5-1d6018de265e',
        'agent_id': '00000000-0000-0000-0000-000000000000',
        'agent_name': '',
        'description': '',
        'display_name': 'aajsSxM@odio_mock_200_account_workmail_67028231-BFC9-352E-
A3D9-0D934164DDC3_edd0ec02-67df-4017-abd8-ee37611dbeb5A',
        'owner_id': '',
        'owner_name': '',
        'resource_id': '',
        'consistent': false,
        'marked_for_deletion': false,
        'last_seen_at': '2025-09-29T15:59:09Z',
        'vault': {
          'id': '21882913-cc56-646e-1036-9eac88c3977a',
          'name': 'pk-customer',
          'storage_type': 'ONLINE'
        }
```

```
    }
  ]
}
```

4. Преобразуйте текст в формате JSON, содержащийся в теле ответа, в объект, а затем извлеките данные об архивах из ответа:

```
>>> archives = response.json()['items']
```

2.10.12 Резервная копия

Резервная копия (точка востановления) – это сохранённые данные ресурса и метаданные для них. Резервная копия создаётся при операциях резервного копирования. Данные резервной копии используются в операциях восстановления.

Все операции с резервными копиями находятся в конечной точке /backups.

2.10.12.1 Структура объекта JSON резервной копии

Описание структуры объекта JSON резервной копии доступно по ссылке.

2.10.12.2 Справочники

Справочники доступны по ссылке.

2.10.12.3 Действия с резервными копиями

Операция	Используемые методы и конечные точки
"Просмотр резервных копий" (стр. 159)	GET /backups
Получить количество хранимых резервных копий в облачном хранилище возможно с помощью операции "Просмотр резервных копий" (стр. 159).	GET /backups

2.10.12.4 Просмотр резервных копий

1. Выполните аутентификацию на платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
```

2. [Необязательно] Задайте переменную filters

```
>>> filters = {
...     'count': true
... }
```

Примечание

Для получения количества хранимых резервных копий, размещенных в облачном хранилище, укажите в запросе параметр count = true. Количество резервных копий будет возвращено в атрибуте total_count.

Список доступных параметров строки запроса приведен в справочнике API.

3. Отправьте запрос GET на конечную точку /backups:

```
>>> response = requests.get(f'{base_url}/backups', headers=auth, params=filters)
```

4. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, тело ответа содержит массив items в формате JSON, содержащий данные резервных копий. После преобразования в объект он будет выглядеть так:

```
>>> pprint.pprint(response.json())
{
  'items': [
    {
      'backup_id': 'b79a69d6-3baf-44d1-b367-5a9038e258f2',
      'vault_id': '21882913-cc56-646e-1036-9eac88c3977a',
      'archive_id': '5b342641-8f24-a094-29f1-97ff33745e42',
      'legacy_id': 'avfs:/online?account%3dpk-
customer%26provider%3dCyberProtect#arl:/00000000-0000-0000-0000-
000000000000/00000000-0000-0000-0000-000000000000/5B342641-8F24-A094-29F1-
97FF33745E42/B79A69D6-3BAF-44D1-B367-5A9038E258F2?archive_
name%3dAALToOq@odio_mock_200_account_workmail_E68393C6-E69E-3765-815F-
E056D6A6F50E_edd0ec02-67df-4017-abd8-ee37611dbeb5A',
      'created_at': '2025-09-29T16:02:45Z',
      'size': 24576,
      'deduplicated_size': 0,
      'backed_up_data_size': 0,
      'original_data_size': 38934,
      'tenant_id': 'f9cf6b51-90c8-43bb-8da5-1d6018de265e',
      'resource_ids': [
        'E68393C6-E69E-3765-815F-E056D6A6F50E'
      ],
      'type': 'FULL',
```

```
        'marked_for_deletion': false,
        'last_seen_at': '2025-09-29T16:02:45Z',
        'validation_status': null,
        'created_in_network_isolation': false,
        'deleted': false
      },
      {
        'backup_id': '2382810e-8197-47ee-85dc-ba3ef28eb98c',
        'vault_id': '21882913-cc56-646e-1036-9eac88c3977a',
        'archive_id': '86150517-97b7-c2e2-0413-f137b16ae87f',
        'legacy_id': 'avfs:/online?account%3dpk-
customer%26provider%3dCyberProtect#arl:/00000000-0000-0000-0000-
000000000000/00000000-0000-0000-0000-000000000000/86150517-97B7-C2E2-0413-
F137B16AE87F/2382810E-8197-47EE-85DC-BA3EF28EB98C?archive_
name%3dAADDcSu@odio_mock_200_account_workmail_A5C0987D-F28C-3B2B-A8DD-
93D833A3FA05_edd0ec02-67df-4017-abd8-ee37611dbeb5A',
        'created_at': '2025-09-29T16:02:45Z',
        'size': 24576,
        'deduplicated_size': 0,
        'backed_up_data_size': 0,
        'original_data_size': 39405,
        'tenant_id': 'f9cf6b51-90c8-43bb-8da5-1d6018de265e',
        'resource_ids': [
          'A5C0987D-F28C-3B2B-A8DD-93D833A3FA05'
        ],
        'type': 'FULL',
        'marked_for_deletion': false,
        'last_seen_at': '2025-09-29T16:02:45Z',
        'validation_status': null,
        'created_in_network_isolation': false,
        'deleted': false
      },
      {
        'backup_id': 'd9a2f2ba-3f36-4be1-92a4-989479bccedc',
        'vault_id': '21882913-cc56-646e-1036-9eac88c3977a',
        'archive_id': 'afe414ad-34c9-1dd0-6955-27619a7ce7a3',
        'legacy_id': 'avfs:/online?account%3dpk-
customer%26provider%3dCyberProtect#arl:/00000000-0000-0000-0000-
000000000000/00000000-0000-0000-0000-000000000000/AFE414AD-34C9-1DD0-6955-
27619A7CE7A3/D9A2F2BA-3F36-4BE1-92A4-989479BCCEDC?archive_
name%3daajsSxM@odio_mock_200_account_workmail_1316D72D-9923-3061-B4DA-
26A46398A795_edd0ec02-67df-4017-abd8-ee37611dbeb5A',
        'created_at': '2025-09-29T16:02:45Z',
        'size': 24576,
        'deduplicated_size': 0,
        'backed_up_data_size': 0,
        'original_data_size': 38920,
        'tenant_id': 'f9cf6b51-90c8-43bb-8da5-1d6018de265e',
        'resource_ids': [
          '1316D72D-9923-3061-B4DA-26A46398A795'
        ],
```

```
            'type': 'FULL',
            'marked_for_deletion': false,
            'last_seen_at': '2025-09-29T16:02:45Z',
            'validation_status': null,
            'created_in_network_isolation': false,
            'deleted': false
        }
        ...
        ],
    "paging": {
        "total_count": 51,
        "cursors": {}
    },
}
```

5. Преобразуйте текст в формате JSON, содержащийся в теле ответа, в объект, а затем извлеките данные о резервных копиях из ответа:

```
>>> backups = response.json()['items']
```

2.10.13 Задача

Задача – это единица выполнения, достаточно значимая (ценная) для того, чтобы конечный пользователь мог отслеживать её и управлять ею. Задача состоит из набора действий (шагов в выполнении задачи), которые выполняются в определённое время или при наступлении определённого события.

Все операции с задачами находятся в конечной точке /tasks.

2.10.13.1 Структура объекта JSON задачи

Описание структуры объекта JSON задачи доступно по ссылке.

2.10.13.2 Справочники

Справочники доступны по ссылке.

2.10.13.3 Действия с задачами

Операция	Используемые методы и конечные точки
"Получение списка задач" (стр. 163)	GET /tasks
"Получение информации об отдельной задаче" (стр. 166)	GET /tasks/{tasks_id}

2.10.13.4 Получение списка задач

1. Выполните аутентификацию на платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   ```

2. Отправьте запрос GET на конечную точку /tasks:

   ```
   >>> response = requests.get(f'{base_url}/tasks', headers=auth)
   ```

   Примечание

   Список доступных параметров строки запроса приведен в справочнике API.

3. Проверьте код состояния запроса:

   ```
   >>> response.status_code
   200
   ```

   Код состояния HTTP 200 означает, что запрос был успешно выполнен.

   Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

   Кроме того, тело ответа содержит массив items в формате JSON, содержащий объекты задачи. После преобразования в объект он будет выглядеть так:

   ```
   >>> pprint.pprint(response.json())
   {
     'items': [
       {
         'cancel_requested': false,
         'cancellable': true,
         'completed_at': '2025-09-23T12:48:24.590984669Z',
         'context': {
           'is_legacy': true,
           'is_process_root': true,
           'persistent': {
             'id': '8b6e876b-69b3-4b2b-95a9-077722b1fe8d',
             'name': 'main-win11',
             'owner_id': '18'
           },
           'specific': 'Business',
           'title': 'Adding agent 'main-win11' to the management server',
           'user_name': 'main-win11\\CMS User'
         },
   ```

```
    'enqueued_at': '2025-09-23T12:48:24.590984669Z',
    'id': '1628761130376499200',
    'issuer': {
      'cluster_id': '',
      'id': ''
    },
    'priority': 'NORMAL',
    'queue': 'legacySync',
    'result': {
      'code': 'OK',
      'payload': 'MachineManagement::AddAgentResponse'
    },
    'started_at': '2025-09-23T12:48:24.590984669Z',
    'started_by_user': 'main-win11\\CMS User',
    'state': 'COMPLETED',
    'tenant': {
      'id': 'f9cf6b51-90c8-43bb-8da5-1d6018de265e',
      'locator': '/1/17/18/',
      'name': 'pk-customer'
    },
    'type': 'A59E8BF2-39C3-42C4-B667-CB672381A214',
    'updated_at': '2025-09-23T12:48:24.593232809Z',
    'uuid': '65d8ffd7-d953-49d9-8d7d-1fdc3714d3ae'
  },
  {
    'cancel_requested': false,
    'cancellable': true,
    'completed_at': '2025-09-23T12:48:22Z',
    'context': {
      '_runtime': {
        'source_stamp': 20
      },
      'command_id': 'E510638A-E988-4D90-9985-80CFAF8981FC',
      'execution_queue': 'default_queue',
      'has_children': false,
      'is_legacy': true,
      'is_process_root': true,
      'machine_name': 'main-win11',
      'message_id': '',
      'parent_uuid': 'f14e531d-faa2-49b9-a250-a3a05a48afb7',
      'process_id': 76332,
      'specific': 'Business',
      'title': 'Upgrading Startup Recovery Manager',
      'user_name': 'WORKGROUP\\MAIN-WIN11$'
    },
    'enqueued_at': '2025-09-23T12:48:21Z',
    'executor': {
      'cluster_id': '',
      'id': '8b6e876b-69b3-4b2b-95a9-077722b1fe8d'
    },
    'id': '1628761138328391680',
```

```
      'issuer': {
        'cluster_id': '',
        'id': ''
      },
      'priority': 'NORMAL',
      'progress': {
        'current': 100,
        'total': 100
      },
      'queue': 'legacySync',
      'result': {
        'code': 'OK',
        'payload': null
      },
      'started_at': '2025-09-23T12:48:21Z',
      'state': 'COMPLETED',
      'tenant': {
        'id': 'f9cf6b51-90c8-43bb-8da5-1d6018de265e',
        'locator': '/1/17/18/',
        'name': 'pk-customer'
      },
      'type': 'E510638A-E988-4D90-9985-80CFAF8981FC',
      'updated_at': '2025-09-23T12:48:26.58076431Z',
      'uuid': 'f14e531d-faa2-49b9-a250-a3a05a48afb7'
    },
    {
      'cancel_requested': false,
      'cancellable': true,
      'completed_at': '2025-09-23T12:52:37.4201818Z',
      'context': {
        'account_name': 'VK WorkMail'
      },
      'enqueued_at': '2025-09-23T12:52:37.1783328Z',
      'executor': {
        'cluster_id': '',
        'id': '8b6e876b-69b3-4b2b-95a9-077722b1fe8d'
      },
      'id': '1628762140825767936',
      'issuer': {
        'cluster_id': '',
        'id': ''
      },
      'priority': 'NORMAL',
      'progress': {
        'current': 100,
        'total': 100
      },
      'queue': 'queue_gba_discovery',
      'result': {
        'code': 'OK',
        'payload': {
```

```
                'accounts': 0,
                'domains': 0,
                'start_time': '2025-09-23T15:52:37.2245779+03:00'
              }
            },
            'started_at': '2025-09-23T12:52:37.1896586Z',
            'state': 'COMPLETED',
            'tenant': {
              'id': 'f9cf6b51-90c8-43bb-8da5-1d6018de265e',
              'locator': '/1/17/18/',
              'name': 'pk-customer'
            },
            'type': 'GenericDiscovery',
            'updated_at': '2025-09-23T12:52:47.229676264Z',
            'uuid': '04424590-3e69-4f47-a757-8a1fe097078b'
        }
    ],
    'paging': {
        'cursors': {
            'after': 'eyJMb2QiOiJza...'
        }
    },
    'size': 3,
    'time_stamp': '1763130428692733875ns'
}
```

4. Преобразуйте текст в формате JSON, содержащийся в теле ответа, в объект, а затем извлеките список задач из ответа:

```
>>> tasks = response.json()['items']
```

2.10.13.5 Получение информации об отдельной задаче

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
```

2. Присвойте переменной task_id UUID задачи, найденной с помощью поиска (см. раздел "Получение списка задач" (стр. 163)). В данном примере использован идентификатор первой задачи:

```
>>> task_id = tasks[0]['id']
>>> task_id
'1628761130376499200'
```

3. Отправьте запрос GET на конечную точку /tasks/<task_id>:

```
>>> response = requests.get(f'{base_url}/tasks/{task_id}', headers=auth)
```

Примечание

Список доступных параметров строки запроса приведен в справочнике API.

4. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, тело ответа в формате JSON содержит информацию о задаче. После преобразования в объект он будет выглядеть так:

```
>>> pprint.pprint(response.json())
{
  'cancel_requested': false,
  'cancellable': true,
  'completed_at': '2025-09-23T12:48:24.590984669Z',
  'context': {
    'is_legacy': true,
    'is_process_root': true,
    'persistent': {
      'id': '8b6e876b-69b3-4b2b-95a9-077722b1fe8d',
      'name': 'main-win11',
      'owner_id': '18'
    },
    'specific': 'Business',
    'title': 'Adding agent 'main-win11' to the management server',
    'user_name': 'main-win11\\CMS User'
  },
  'enqueued_at': '2025-09-23T12:48:24.590984669Z',
  'id': '1628761130376499200',
  'issuer': {
    'cluster_id': '',
    'id': ''
  },
  'priority': 'NORMAL',
  'queue': 'legacySync',
  'result': {
    'code': 'OK',
    'payload': 'MachineManagement::AddAgentResponse'
  },
  'started_at': '2025-09-23T12:48:24.590984669Z',
  'started_by_user': 'main-win11\\CMS User',
  'state': 'COMPLETED',
```

```
    'tenant': {
      'id': 'f9cf6b51-90c8-43bb-8da5-1d6018de265e',
      'name': 'pk-customer'
    },
    'type': 'A59E8BF2-39C3-42C4-B667-CB672381A214',
    'updated_at': '2025-09-23T12:48:24.593232809Z',
    'uuid': '65d8ffd7-d953-49d9-8d7d-1fdc3714d3ae'
}
```

2.10.14 Действие

Действие – это последовательный набор операций, направленных на достижение некоторой конечной и четко определенной цели пользователя. Обычно является шагом выполнения задачи. Действие выполняется для ресурса, его параметры определены используемыми политиками. Действие может быть инициировано пользователем или самой платформой, используется для подробного представления задачи конечному пользователю.

Все операции с действиями находятся в конечной точке /activities.

2.10.14.1 Структура объекта JSON действия

Описание структуры объекта JSON действия доступно по ссылке.

2.10.14.2 Справочники

Справочники доступны по ссылке.

2.10.14.3 Операции с действиями

Операция	Используемые методы и конечные точки
"Получение списка действий" (стр. 168)	GET /activities
"Получение информации о действии" (стр. 170)	GET /activities/{activity_id}

2.10.14.4 Получение списка действий

1. Выполните аутентификацию на платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

   ```
   >>> base_url  # the base URL of the API
   'https://installaddress.ru/api/v1'
   >>> auth  # the 'Authorization' header value with the access token
   {'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
   ```

2. Отправьте запрос GET на конечную точку /activities:

```
>>> response = requests.get(f'{base_url}/activities', headers=auth)
```

Примечание

Список доступных параметров строки запроса приведен в справочнике API.

3. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, тело ответа содержит массив items в формате JSON, содержащий объекты действий. После преобразования в объект он будет выглядеть так:

```
>>> pprint.pprint(response.json())
{
   'items': [
      {
         'completed_at': '2025-11-17T07:38:59.4628885Z',
         'context': {
            'is_legacy': true,
            'is_process_root': true,
            'persistent': {
               'id': 'd45dc397-541a-49fa-a0b1-fb20b93d09d9',
               'name': 'Template-WindowsServer-2022',
               'owner_id': '00000000-0000-0000-0000-000000000000'
            },
            'specific': 'Business',
            'title': 'Adding agent 'Template-WindowsServer-2022' to the management server',
            'user_name': 'Template-WindowsServer-2022\\CMS User'
         },
         'created_at': '2025-11-17T07:38:59.4628885Z',
         'executor': {
            'cluster_id': '',
            'id': ''
         },
         'id': '1647694869855993856',
         'result': {
            'code': 'OK',
            'payload': 'MachineManagement::AddAgentResponse'
         },
         'started_at': '2025-11-17T07:38:59.4628885Z',
         'started_by_user': 'Template-WindowsServer-2022\\CMS User',
         'state': 'COMPLETED',
         'task_id': '1647694869855993856',
         'tenant': {
            'id': '00000000-0000-0000-0000-000000000000',
```

```
            'locator': '/00000000-0000-0000-0000-000000000000/',
            'name': 'Organization'
          },
          'type': 'A59E8BF2-39C3-42C4-B667-CB672381A214',
          'updated_at': '2025-11-17T07:38:59.4634214Z',
          'uuid': '7b6f1aba-1d6c-4d4e-9d5c-fe97316a641a'
        }
    ],
    'size': 1,
    'time_stamp': '1763367418236566900ns'
}
```

4. Преобразуйте текст в формате JSON, содержащийся в теле ответа, в объект, а затем извлеките действия из ответа:

```
>>> activities = response.json()['items']
```

2.10.14.5 Получение информации о действии

1. Выполните аутентификацию на облачной платформе с помощью оболочки Python (см. "Аутентификация на платформе Кибер Бэкап Облачный" (стр. 19)).

   Должны стать доступны следующие переменные:

```
>>> base_url  # the base URL of the API
'https://installaddress.ru/api/v1'
>>> auth  # the 'Authorization' header value with the access token
{'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImMwMD...'}
```

2. Присвойте переменной activity_id UUID задачи, найденной с помощью поиска (см. "Получение списка действий" (стр. 168)). В данном примере использован идентификатор первого действия:

```
>>> activity_id = activities[0]['id']
>>> activity_id
'1647694869855993856'
```

3. Отправьте запрос GET на конечную точку /activities/<activity_id>:

```
>>> response = requests.get(f'{base_url}/activities/{activity_id}', headers=auth)
```

Примечание

Список доступных параметров строки запроса приведен в справочнике API.

4. Проверьте код состояния запроса:

```
>>> response.status_code
200
```

Код состояния HTTP 200 означает, что запрос был успешно выполнен.

Другой код состояния означает, что произошла ошибка. Подробные сведения об ошибке см. в разделе "Коды состояния HTTP" (стр. 27).

Кроме того, тело ответа в формате JSON содержит информацию о действии. После преобразования в объект она будет выглядеть так:

```
>>> pprint.pprint(response.json())
{
   'completed_at': '2025-11-17T07:38:59.4628885Z',
   'context': {
      'is_legacy': true,
      'is_process_root': true,
      'persistent': {
         'id': 'd45dc397-541a-49fa-a0b1-fb20b93d09d9',
         'name': 'Template-WindowsServer-2022',
         'owner_id': '00000000-0000-0000-0000-000000000000'
      },
      'specific': 'Business',
      'title': 'Adding agent 'Template-WindowsServer-2022' to the management server',
      'user_name': 'Template-WindowsServer-2022\\CMS User'
   },
   'created_at': '2025-11-17T07:38:59.4628885Z',
   'executor': {
      'cluster_id': '',
      'id': ''
   },
   'id': '1647694869855993856',
   'result': {
      'code': 'OK',
      'payload': 'MachineManagement::AddAgentResponse'
   },
   'started_at': '2025-11-17T07:38:59.4628885Z',
   'started_by_user': 'Template-WindowsServer-2022\\CMS User',
   'state': 'COMPLETED',
   'task_id': '1647694869855993856',
   'tenant': {
      'id': '00000000-0000-0000-0000-000000000000',
      'locator': '/00000000-0000-0000-0000-000000000000/',
      'name': 'Organization'
   },
   'type': 'A59E8BF2-39C3-42C4-B667-CB672381A214',
   'updated_at': '2025-11-17T07:38:59.4634214Z',
   'uuid': '7b6f1aba-1d6c-4d4e-9d5c-fe97316a641a'
}
```

2.11 Справочники

2.11.1 Справочник API

Справочник находится по ссылке.

2.11.2 Справочник по атрибутам ресурсов

Справочник находится по ссылке.

Тип ресурса	Ссылка на описание параметров
Физическая машина, виртуальная машина, группы	Параметры описаны в схеме Resource
Ресурсы Microsoft SQL Server	Параметры описаны в схеме MssqlResource
Ресурсы Microsoft Exchange Server	Параметры описаны в схеме MsexchangeResource

2.11.3 Справочник по настройкам политик

Наименование политики	Ссылка на описание параметров
policy.backup.machine	Резервное копирование на уровне диска (вся машина) данных физических и виртуальных машин (в том числе VMWare ESXi, Hyper-V, OpenStack, KVM, Citrix, Nutanix). Параметры описаны в схеме MachineBackupSettings
policy.backup.image	Резервное копирование на уровне диска (отдельные диски и тома) данных физических и виртуальных машин. Параметры описаны в схеме DiskBackupSettings
policy.backup.mssql	Резервное копирование баз данных Microsoft SQL Server. Параметры описаны в схеме MssqlBackupSettings
policy.backup.msexchange.db	Резервное копирование баз данных Microsoft Exchange Server. Параметры описаны в схеме MsexchangeDbBackupSettings
policy.backup.msexchange.doc	Резервное копирование почтовых ящиков Microsoft Exchange Server. Параметры описаны в схеме MsexchangeDocBackupSettings

2.12 Список изменений

Список изменений находится по ссылке.

3 Интеграция с агентами

3.1 Параметры командной строки для установщика агентов

3.1.1 Параметры программы установки для агентов Windows

```
--help
```

Показать страницу со справочной информацией.

```
--quiet
```

Запускает программу установки без отображения интерфейса пользователя (автоматическая установка).

```
--log-dir=<путь>
```

Путь к папке для сохранения журналов установки.

```
--language=<code>
```

Язык, который будет использовать в продукте и его компонентах. Поддерживаемые значения:

- en – English
- ru – Русский

```
--register-only
```

Применимо только к Кибер Бэкап Облачный. Пропускает установку и показывает только страницу регистрации (регистрация OAuth 2.0). Не использовать с параметром --quiet.

```
--anti-tamper-password=<пароль>
```

Пароль с защитой от несанкционированного изменения. Может потребоваться изменить некоторые установленные компоненты.

3.1.2 Файлы установки

```
--installation-files=<путь/url-адрес>
```

URL-адрес или путь файловой системы к файлам установки.

```
--skip-https-cert-verification
```

Пропустить проверку сертификата сервера при скачивании установочных файлов.

```
--skip-file-trust-verification
```

Пропустить проверку доверия скачанных файлов установки.

```
--attempts-number=<число>
```

Количество попыток скачать файл установки. По умолчанию установлено значение «1».

```
--cache-directory=<tmp>
```

Путь к локальной папке, в которой перед установкой будет создан кэш файлов установки.

3.1.3 Компоненты

```
--add-components=<компонент1,компонент2,...,компонентN>
```

Компоненты и группы компонентов для добавления в устанавливаемый продукт.

Если этот параметр не указан, продукт устанавливается с набором компонентов по умолчанию в зависимости от настройки программы установки. Если продукт уже установлен, он будет восстановлен или обновлен в соответствии с версией программы установки. Если этот параметр указан, продукт будет установлен с компонентами и группами компонентов, указанными в значении (можно объединить компоненты и группы в значение). Указанные компоненты, которые уже установлены, будут восстановлены или обновлены в соответствии с версией программы установки.

Поддерживаемые значения компонента, общие для Кибер Бэкап и Кибер Бэкап Облачный :

- agentForAd – Агент для Active Directory;
- agentForExchange – Агент для Exchange;
- agentForHyperV – Агент для Hyper-V;
- agentForEsx – Агент для VMware (Windows);
- agentForOracle – Агент для Oracle;
- agentForSql – Агент для SQL;
- agentForPostgreSql – Агент для PostgreSQL;
- agentForWindows – Агент для Windows;
- commandLine – Программа командной строки;
- mediaBuilder – Мастер создания загрузочных носителей;
- trayMonitor – Мониторинг Защиты Данных.

Поддерживаемые значения компонента только для Кибер Бэкап:

- managementServer – Сервер управления;
- remoteInstallation – Компоненты для удаленной установки;
- virtualAppliance – Агент для VMware (виртуальное устройство);
- agentForOffice365 – Агент для Office 365;
- pxeServer – PXE-сервер;
- agentForCommunigate – Агент для CommuniGate Pro;
- agentForWorkmail – Агент для VK WorkMail;
- storageNode – Узел хранения;
- catalogService – Служба каталога.

Поддерживаемые значения группы:

- all – В этой группе объединены все компоненты;
- allAgents – В этой группе объединены все агенты.

```
--remove-components=<компонент 1,компонент 2,...,компонент>
```

Компоненты и группы компонентов для удаления.

Используя этот параметр, можно удалить только компоненты продукта. Чтобы полностью удалить продукт, откройте "Панель управления" и выберите "Программы и компоненты" > "Cyber Backup" > "Удалить". Поддерживаемые значения см. в описании параметра -add.

3.1.4 Путь установки

```
--installdir=<путь>
```

Путь к папке для установки продукта и его компонентов. Если папка не существует, она будет создана.

3.1.5 Регистрация

```
--reg-address=<url>
```

URL-адрес сервера управления.

Для Кибер Бэкап возможно также использование <имя/ip-адрес>.

```
--reg-login=<имя входа>
```

Имя входа для сервера управления. Не используйте с --reg-token. Если этот параметр указан, параметр --registration необязателен.

```
--reg-password=<пароль>
```

Пароль для сервера управления. Не используйте с --reg-token. Если этот параметр указан, параметр --registration необязателен.

```
--reg-token=<маркер>
```

Маркер регистрации. Не используйте с параметрами --reg-login или --reg-password. Если этот параметр указан, параметр –registration необязателен.

```
--registration=[skip | by-credentials | by-token | device-flow | current-user]
```

Тип регистрации агента после установки. Чтобы пропустить регистрацию, укажите параметр skip. Чтобы зарегистрировать агенты по учетным данным или маркеру, укажите by-credentials или by-token и введите соответствующие параметры.

Применимо только к Кибер Бэкап Облачный. Чтобы зарегистрировать агентов, используя протокол OAuth 2.0, укажите параметр device-flow. В этом случае программа установки покажет страницу регистрации по окончании установки. Не используйте --registration=device-flow с параметром --quiet.

Применимо только к Кибер Бэкап. Чтобы зарегистрировать агентов с использованием учетной записи текущего пользователя Windows, укажите current-user.

```
--reg-tenant=<клиент>
```

Применимо только к Кибер Бэкап. Отдел для регистрации агента.

```
--reg-transport=[https | https-ca-system | https-ca-bundle | https-pinned-public-key]
```

Применим только для Кибер Бэкап. Тип передачи данных, который будет использоваться для регистрации агента. Чтобы выполнить регистрацию через HTTPS без проверки сертификата, укажите https. Чтобы выполнить регистрацию через HTTPS с проверкой сертификата в системном центре сертификации, укажите https-ca-system. Чтобы выполнить регистрацию через HTTPS с проверкой сертификата с использованием пакета центра сертификации, который прилагается к продукту, укажите https-ca-bundle. Чтобы выполнить регистрацию по HTTPS с проверкой сертификата с использованием закрепленного открытого ключа, укажите https-pinned-public-key. В этом случае необходимо указать параметр reg-transport-pinned-public-key.

```
--reg-transport-pinned-public-key=<закрепленный открытый ключ>
```

Применим только для Кибер Бэкап. Закрепленный открытый ключ. Если этот параметр определен, параметр reg-transport должен иметь значение https-pinned-public-key или быть опущен.

```
--reg-script=<путь к файлу скрипта регистрации>
--reg-script-params="-c <путь к файлу registration.json> -u <имя пользователя администратора Active Directory> -p <пароль администратора Active Directory> -l <путь к log файлу>"
```

3.1.6 Учетная запись для входа службы агента

```
--agent-account-login=<имя входа>
```

Имя входа для учетной записи службы.

```
--agent-account-password=<пароль>
```

Пароль для учетной записи службы.

```
--agent-account=[system | new | custom]
```

Учетная запись, с которой будет запускаться служба агента Кибер Бэкап Облачный. Чтобы использовать учетные записи пользователя услуги, укажите system. Чтобы создать новую учетную запись, укажите new. Чтобы использовать существующую учетную запись, укажите custom или оставьте это поле пустым.

3.1.7 Учетная запись входа для службы сервера управления

Параметры раздела применимы только для Кибер Бэкап.

```
--management-server-account-login=<имя входа>
```

Имя входа для учетной записи службы.

```
--management-server-account-password=<пароль>
```

Пароль для учетной записи службы.

```
--management-server-account=[system | new | custom]
```

Учетная запись, с которой будет запускаться служба сервера управления. Чтобы использовать учетные записи пользователя услуги, укажите system. Чтобы создать новую учетную запись, укажите new. Чтобы использовать существующую учетную запись, укажите custom или оставьте это поле пустым.

3.1.8 Учетная запись входа для службы узла хранения

Параметры раздела применимы только для Кибер Бэкап.

```
--storage-mode-account-login=<имя входа>
```

Имя входа для учетной записи службы.

```
--storage-mode-account-password=<пароль>
```

Пароль для учетной записи службы.

```
--storage-node-account=[system | new | custom]
```

Учетная запись, с которой будет запускаться служба узла хранения. Чтобы использовать учетные записи пользователя услуги, укажите system. Чтобы создать новую учетную запись, укажите new. Чтобы использовать существующую учетную запись, укажите custom или оставьте это поле пустым.

3.1.9 Порт HTTP

Параметры раздела применимы только для Кибер Бэкап.

```
--web-server-port=<порт>
```

Этот порт используется для безопасного обмена данными с веб-интерфейсом.

3.1.10 TCP-порт для компонентов

Параметры раздела применимы только для Кибер Бэкап.

```
--zmq-port=<порт>
```

Этот порт используется для обмена данными между компонентами продукта.

3.1.11 Прокси-сервер HTTP

```
--http-proxy-address=<хост>:<порт>
```

Имя/IP-адрес и порт настраиваемого прокси-сервера HTTP.

```
--http-proxy-login=<имя входа>
```

Имя входя для настраиваемого прокси-сервера HTTP.

```
--http-proxy-password=<пароль>
```

Пароль для настраиваемого прокси-сервера HTTP.

```
--http-proxy=[none | system | custom]
```

Дает инструкцию, использовать ли прокси-сервер HTTP для резервного копирования данных в облачное хранилище данных и восстановление данных из облачного хранилища по сети Интернет. Если не нужно использовать прокси-сервер, укажите none. Чтобы использовать прокси-сервер в масштабе системы, укажите system. Чтобы использовать какой-либо другой прокси-сервер, укажите custom, а затем укажите его адрес и учетные данные в соответствующих ключах.

3.1.12 vCenter/Esxi

```
--esxi-address=<хост>
```

Имя хоста или IP-адрес vCenter Server или ESXi.

```
--esxi-login=<имя входа>
```

Имя входа для vCenter Server или ESXi.

```
--esxi-password=<пароль>
```

Пароль для vCenter Server или ESXi.

3.1.13 База данных для сервера управления

Параметры раздела применимы только для Кибер Бэкап.

```
--management-server-db-auth=[service-account | <user>:<password>]
```

Укажите способ проверки подлинности, который будет использоваться для данного экземпляра.

```
--management-server-db-driver=<драйвер>
```

Драйвер для Microsoft SQL Server.

```
--management-server-db-instance=<экземпляр>
```

Экземпляр Microsoft SQL Server.

```
--management-server-db=[built-in | mssql]
```

База данных, которая будет использоваться сервером управления.

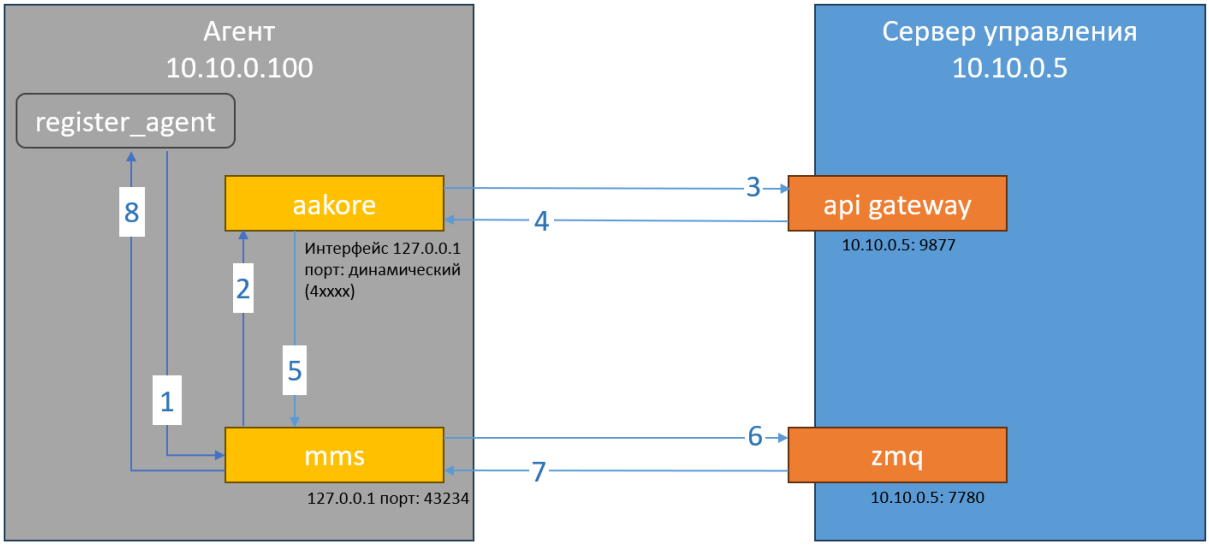
3.1.14 Параметры установки для агентов Linux

Для Кибер Бэкап Облачный см. описание в разделе "Параметры автоматической установки или автоматического удаления" инструкции пользователя.

3.2 Параметры командной строки для регистрационной утилиты

3.2.1 Регистрация агента

Регистрация агента – это процедура подключения агента к серверу управления, который будет монопольно управлять данным агентом.



Регистрация состоит из нескольких этапов:

1. Утилита **register-agent** обращается к службе агента **mms** (127.0.0.1:43234) с просьбой зарегистрировать агента на сервере управления 10.10.0.5 (пример).
2. Служба **mms** передает запрос на службу **aakore** (127.0.0.1:<порт динамический в диапазоне 40000-50000, меняется при перезапуске>).
3. Служба **aakore** через внешний интерфейс обращается к службе **api-gateway** (10.10.0.5:9877) на сервере управления для обмена ключами и первичной регистрации агента.
4. Служба **api-gateway** регистрирует агента и выдает ключи службе **aakore** для дальнейшей работы по установленному ранее соединению.
5. Служба **aakore** передает полученные ключи службе **mms** (127.0.0.1:43234).
6. Служба **mms** устанавливает соединение со службой **zmq** (10.10.0.5:7780) на сервере управления и завершает регистрацию. На этом этапе в разделе "Настройки - агенты" заполняется информация об агенте (ОС, IP-адреса, версия агента и т. д.), и он становится активен.
7. Начинается постоянная синхронизация агента и сервера управления.
8. Служба **mms** возвращает результат регистрации утилите **register-agent**.

3.2.2 Ручная регистрация

В случае если Вам необходимо зарегистрировать уже установленного агента на новом сервере управления или обновить его регистрацию, то можно воспользоваться одним из следующих способов.

3.2.2.1 Регистрация агента для Windows

Обычная регистрация агента, если разрешена анонимная регистрация

На компьютере с агентом откройте командную строку: **Пуск** -> **Поиск** (или **Выполнить**), введите cmd.exe и нажмите **Enter**.

Перейдите в папку C:\Program Files\Acronis\RegisterAgentTool\:

```
cd "C:\Program Files\Acronis\RegisterAgentTool"
```

Выполните команду:

```
register_agent.exe -o register -a <адрес сервера управления>
```

Регистрация с указанием пользователя

```
register_agent.exe -o register -a <адрес сервера управления> -u <имя пользователя> -p <пароль>
```

В приведенной выше команде:

<адрес сервера управления> – это IP-адрес или имя хоста сервера управления, на котором должен быть зарегистрирован агент.

<username> – это имя администратора для сервера управления.

<пароль> – пароль администратора для сервера управления.

Пример:

```
register_agent.exe -o register -a 192.168.1.0 -u admin -p admin1234
```

Регистрация агента в определенном отделе

```
register_agent.exe -o register -a <адрес сервера управления> --tenant <id_отдела> -u <имя пользователя> -p <пароль>
```

Чтобы найти идентификатор отдела, в консоли управления сервера управления перейдите в **Настройки** -> **Учетные записи** и далее в нужный Отдел. После этого нажмите кнопку **Сведения** и скопируйте значение свойства "id" (значение вида "556c6428-34ef-11ed-83ee-00155d8b5d06").

Регистрация агента с помощью токена

```
register_agent.exe -o register -a <адрес сервера управления:порт> --token <токен>
```

<токен> – маркер регистрации который можно создать в разделе "Устройства" -> "Добавить" -> "Маркер регистрации - создать".

Удаление регистрации агента

Для удаления регистрации агента используйте команду регистрации с параметром -o unregister:

```
register_agent.exe -o unregister
```

3.2.2.2 Регистрация агента для Linux

Обычная регистрация

На машине агента откройте **Терминал**.

Выполните команду:

```
/usr/lib/Acronis/RegisterAgentTool/RegisterAgent -o register -a <адрес сервера управления>
```

Регистрация с указанием пользователя

```
/usr/lib/Acronis/RegisterAgentTool/RegisterAgent -o register -a <адрес сервера управления> -u <имя пользователя> -p <пароль>
```

В приведенной выше команде:

<адрес сервера управления> – это IP-адрес или имя хоста сервера управления, на котором должен быть зарегистрирован агент.

<username> – это имя пользователя администратора для сервера управления.

<пароль> – пароль администратора для сервера управления.

Регистрация агента с помощью токена регистрации

```
/usr/lib/Acronis/RegisterAgentTool/RegisterAgent -o register -a <адрес сервера управления> --token <токен>
```

В приведенной выше команде:

<токен> – маркер регистрации, который можно создать в разделе "Устройства" -> "Добавить" -> "Маркер регистрации - создать".

3.2.2.3 Добавить виртуальное устройство (Virtual appliance)

Откройте консоль устройства в консоли гипервизора, затем нажмите Ctrl + Shift + F2, чтобы открыть терминал на устройстве.

Обычная регистрация

Выполните команду:

```
/bin/register_agent -o register -a <адрес сервера управления>
```

Регистрация с указанием пользователя

```
/bin/register_agent -o register -a <адрес сервера управления> -u <имя пользователя> -p
<пароль>
```

В приведенной выше команде:

- <адрес сервера управления> – это IP-адрес или имя хоста сервера управления, на котором должен быть зарегистрирован агент;
- <username> – это имя пользователя администратора для сервера управления;
- <пароль> – пароль администратора для сервера управления.

Регистрация агента с помощью токена регистрации

```
/bin/register_agent -o register -a <адрес сервера управления:порт> --token <токен>
```

В приведенной выше команде:

<токен> – маркер регистрации, который можно создать в разделе "Устройства" -> "Добавить" -> "Маркер регистрации - создать".

Глоссарий

Указатель

П

Р

С

Т

У

Ч