rustack-esu

Development Team

сент. 13, 2023

Управление ресурсами

1	Установка	3
2	Начало работы	5
	Алфавитный указатель	47

Данная документация описывает использование библиотеки **rustack-esu**, выполняющей роль высокоуровневой абстракции к REST API облака гиперконвергентного решения РУСТЭК-ЕСУ. Описание методов API можно найти в [отдельной документации](#).


```
$ pip install rustack-esu
```


Для взаимодействия с облаком через библиотеку потребуется заранее получить токен доступа. Это можно сделать в панели управления или вызвав соответствующий метод API.

Данный токен необходимо передавать как параметр **token** в конструкторы объектов и в метод **get_object**. Если токен не будет передан, он будет взят из переменной окружения **ESU_API_TOKEN**.

2.1 Manager

Позволяет получить списки всех объектов, доступных текущему пользователю.

```
class esu.Manager(*args, token: Optional[str] = None, **kwargs)
```

Параметры

token (*str*) – Токен для доступа к API. Если не передан, будет использована переменная окружения **ESU_API_TOKEN**

```
get_all_clients()
```

Возвращает список объектов всех доступных пользователю клиентов. Если текущему пользователю был предоставлен доступ к еще одному клиенту, данный список будет содержать два элемента.

Результат

Список объектов *esu.Client*

Тип результата

list

```
get_all_projects()
```

Возвращает список объектов всех доступных пользователю проектов. Если текущий пользователь имеет несколько проектов или ему предоставили доступ к стороннему проекту, данный список будет содержать их все.

Результат

Список объектов *esu.Project*

Тип результата

list

get_all_vdcs()

Возвращает список объектов всех доступных пользователю ВЦОДов. Если текущий пользователь имеет несколько ВЦОДов или ему был предоставлен доступ к сторонним проектам, данный список будет содержать их все.

Результат

Список объектов *esu.Vdc*

Тип результата

list

get_all_vms()

Возвращает список объектов всех доступных пользователю виртуальных выделенных серверов. Если текущий пользователь имеет несколько виртуальных серверов или ему был предоставлен доступ к сторонним проектам, данный список будет содержать их все.

Результат

Список объектов *esu.Vm*

Тип результата

list

get_all_storage_profiles()

Возвращает список объектов всех доступных пользователю профилей хранения.

Результат

Список объектов *esu.StorageProfile*

Тип результата

list

get_all_platforms()

Возвращает список объектов всех доступных пользователю платформ.

Результат

Список объектов *esu.Platform*

Тип результата

list

get_all_firewall_templates()

Возвращает список объектов всех доступных пользователю шаблонов брандмауэра.

Результат

Список объектов *esu.FirewallTemplate*

Тип результата

list

get_all_networks()

Возвращает список объектов всех доступных пользователю сетей.

Результат

Список объектов *esu.Network*

Тип результата
list

2.2 Client

Клиент создается при регистрации пользователя. Пользователям доступен как минимум один клиент или больше в случае, когда пользователю был предоставлен доступ к другому клиенту.

2.2.1 Объект «клиент»

```
class esu.Client(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор
- `name (str)` – Имя
- `payment_model (str)` – Модель взаиморасчетов. **prepay** или **postpay**
- `balance (float)` – Баланс
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения **ESU_API_TOKEN**

Предупреждение: Объект доступен только для чтения и не может быть создан, изменен или удален.

```
classmethod get_object(id, token=None)
```

Получить объект клиента по его ID

Параметры

- `id (str)` – Идентификатор клиента
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения **ESU_API_TOKEN**

Результат

Возвращает объект клиента *esu.Client*

Тип результата

object

```
get_projects()
```

Получить проекты данного клиента.

Результат

Список объектов *esu.Project*

Тип результата

list

2.2.2 Примеры использования

Получить список проектов на определенном клиенте:

```
from esu import Client

client = Client.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
for project in client.get_projects():
    print(f'Проект "{project.name}"')
```

2.3 Project

Объект проекта. Проекты представляют из себя логические сущности, в которые объединены те или иные облачные услуги. Проекты могут создаваться клиентами, а после регистрации всегда существует проект по умолчанию.

2.3.1 Объект «проект»

```
class esu.Project(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор
- `name (str)` – Имя
- `client (object)` – Объект класса `esu.Client`. Клиент, к которому относится проект
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Примечание: Поля `name` и `client` необходимы для создания.

Поле `name` может быть изменено для существующего объекта.

```
classmethod get_object(id, token=None)
```

Получить объект проекта по его ID

Параметры

- `id (str)` – Идентификатор проекта
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект проекта `esu.Project`

Тип результата

object

```
create()
```

Создать объект

Исключение

`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует

`save()`

Сохранить изменения

Исключение

`ObjectHasNoId` – Если производится попытка сохранить несуществующий объект

`destroy()`

Удалить объект

Исключение

`ObjectHasNoId` – Когда производится попытка удалить несуществующий объект

`get_vdcs()`

Получить ВЦОДы в данном проекте. Вернет список объектов *esu.Vdc*.

Результат

Список объектов *esu.Vdc*

Тип результата

list

`get_available_hypervisors()`

Получить список доступных гипервизоров в этом проекте. Вернет список объектов *esu.Hypervisor*.

Результат

Список объектов *esu.Hypervisor*

Тип результата

list

`get_dns_zones()`

Получить список доступных доменных зон в этом проекте. Вернет список объектов *esu.Dns*.

Результат

Список объектов *esu.Dns*

Тип результата

list

`get_k8s_clusters()`

Получить список доступных кластеров Kubernetes в этом проекте. Вернет список объектов *esu.Kubernetes*.

Результат

Список объектов *esu.Kubernetes*

Тип результата

list

`get_s3_storages()`

Получить список доступных s3 хранилищ в этом проекте. Вернет список объектов *esu.S3*.

Результат

Список объектов *esu.S3*

Тип результата

list

2.3.2 Примеры использования

Создать проект на определенном клиенте:

```
from esu import Client, Project

client = Client.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
project = Project(client=client, name='Новый проект')
project.create()
```

Переименовать первый проект на клиенте:

```
from esu import Client, Project

client = Client.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
project = client.get_projects()[0]
project.name = 'Новое имя проекта'
project.save()
```

2.4 Hypervisor

Объект гипервизора. Не имеет самостоятельных методов, используется только при создании ВЦОДа. Может быть получен из метода `esu.Project.get_available_hypervisors()`

2.4.1 Объект «гипервизор»

```
class esu.Hypervisor(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор
- `name (str)` – Имя
- `type (str)` – Тип гипервизора. vmware или kvm

2.4.2 Примеры использования

См. пример [здесь](#)

2.5 Vdc

Объект виртуального центра обработки данных (ВЦОД). ВЦОДы включают в себя ИТ-инфраструктуру, вычислительные ресурсы и ресурсы для хранения информации.

При создании ВЦОД автоматически создаются сеть `esu.Network` и маршрутизатор `esu.Router`, которые не могут быть удалены.

Поддерживается два типа гипервизора: **VMware vSphere ESXi** и **OpenStack KVM**.

2.5.1 Объект «ВЦОД»

```
class esu.Vdc(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор ВЦОД
- `name (str)` – Имя ВЦОД
- `hypervisor (object)` – Объект класса `esu.Hypervisor`
- `project (object)` – Объект класса `esu.Project`. Проект, к которому относится данный ВЦОД
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Примечание: Поля `name`, `hypervisor` и `project` необходимы для создания.

Поле `name` может быть изменено для существующего объекта.

```
classmethod get_object(id, token=None)
```

Получить объект ВЦОД по его ID

Параметры

- `id (str)` – Идентификатор ВЦОД
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект ВЦОД `esu.Vdc`

Тип результата

`object`

```
create()
```

Создать объект

Исключение

`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует

```
save()
```

Сохранить изменения

Исключение

`ObjectHasNoId` – Если производится попытка сохранить несуществующий объект

```
destroy()
```

Удалить объект

Исключение

`ObjectHasNoId` – Когда производится попытка удалить несуществующий объект

```
get_vms()
```

Получить список виртуальных машин, доступных в рамках данного ВЦОД.

РезультатСписок объектов *esu.Vm***Тип результата**

list

get_templates()

Получить список шаблонов ОС для создания виртуальных машин, доступных в данном ВЦОДе.

РезультатСписок объектов *esu.Template***Тип результата**

list

get_storage_profiles()

Получить список профилей хранения, которые используются при добавлении дисков, доступных в данном ВЦОДе.

РезультатСписок объектов *esu.StorageProfile***Тип результата**

list

get_firewall_templates()

Получить список шаблонов брандмауэра, доступных в данном ВЦОДе.

РезультатСписок объектов *esu.FirewallTemplate***Тип результата**

list

get_networks()

Получить список сетей, которые существуют в рамках данного ВЦОД.

РезультатСписок объектов *esu.Network***Тип результата**

list

get_routers()

Получить список маршрутизаторов, которые доступны в рамках данного ВЦОД.

РезультатСписок объектов *esu.Router***Тип результата**

list

get_ports()

Получить список подключений, которые существуют в данном ВЦОДе.

РезультатСписок объектов *esu.Port***Тип результата**

list

`get_disks()`

Получить список дисков, которые существуют в данном ВЦОДе.

Результат

Список объектов *esu.Disk*

Тип результата

list

`create_vm(name, template, password)`

Быстрый способ создать виртуальный сервер в сети по-умолчанию и с настройками по-умолчанию.

Параметры

- `name (str)` – Название нового виртуального сервера
- `template (str)` – Название шаблона системы
- `password (str)` – Пароль, который будет установлен на сервер

`get_k8s_templates()`

Получить список шаблонов k8s для создания кластеров, доступных в данном ВЦОДе.

Результат

Список объектов *esu.KubernetesTemplate*

Тип результата

list

`get_images()`

Получить список образов, доступных в данном ВЦОДе.

Результат

Список объектов *esu.Image*

Тип результата

list

`get_backups()`

Получить список задач резервного копирования, доступных в данном ВЦОДе.

Результат

Список объектов *esu.Backup*

Тип результата

list

2.5.2 Примеры использования

Создание нового ВЦОД VMware в первом доступном пользователю проекте:

```
from esu import Manager, Vdc

project = Manager().get_all_projects()[0]
hypervisor = next(h for h in project.get_available_hypervisors() \
    if h.type == 'vmware')

vdc = Vdc(name='Новый ВЦОД', hypervisor=hypervisor, project=project)
vdc.create()
```

Вывести на экран список доступных в определенном ВЦОДе шаблонов операционных систем:

```
from esu import Manager, Vdc

project = Manager().get_all_projects()[0]
vdc = Vdc.get_object('e5d9a192-c5da-485a-b134-1b14ec9c57d9')
for template in vdc.get_templates():
    print(template.name)
```

2.6 Vm

Виртуальный (выделенный) сервер эмулирует работу отдельного физического сервера. Сервер может быть создан с операционной системой семейства Linux или Windows, может иметь несколько сетевых подключений и дисков.

2.6.1 Объект «виртуальный сервер»

```
class esu.Vm(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор
- `name (str)` – Имя
- `description (str)` – Описание. Любой произвольный пользовательский текст
- `cpu (int)` – Количество ядер
- `ram (int)` – Количество ОЗУ в ГБ
- `power (bool)` – Текущее состояние питания. Включен или выключен
- `vdc (object)` – Объект класса `esu.Vdc`. ВЦОД, к которому относится данный виртуальный сервер
- `template (object)` – Объект класса `esu.Template`. Шаблон операционной системы
- `metadata (list)` – Список объектов класса `esu.VmMetadata`. Список полей, необходимых для создания виртуального выделенного сервера. Например, пароль или имя пользователя.
- `ports (list)` – Список объектов класса `esu.Port`. Список сетей, к которым подключен данный виртуальный сервер
- `disks (list)` – Список объектов класса `esu.Disk`. Список дисков, подключенных к данному виртуальному серверу
- `floating (object)` – Объект класса `esu.Port`. Порт подключения виртуального выделенного сервера к внешней сети. Если `None`, сервер не имеет подключения к внешней сети.
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Примечание: Поля `name`, `cpu`, `ram`, `template`, `ports`, `disks` и `vdc` необходимы для создания.

Поля `metadata`, `description` и `floating` опциональны при создании.

Поля `name`, `description`, `cpu`, `ram`, `floating` могут быть изменены для существующего объекта.

`classmethod get_object(id, token=None)`

Получить объект виртуального сервера по его ID

Параметры

- `id (str)` – Идентификатор виртуального сервера
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект виртуального сервера `esu.Vm`

Тип результата

`object`

`create()`

Создать объект

Исключение

`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует

`save()`

Сохранить изменения

`destroy()`

Удалить объект

Исключение

`ObjectHasNoId` – Когда производится попытка удалить несуществующий объект

`add_disk(disk)`

Создать и присоединить к виртуальному серверу новый диск

Параметры

`disk (object)` – Объект диска `esu.Disk`

`attach_disk(disk)`

Присоединить существующий во ВЦОДе диск к виртуальному серверу

Параметры

`disk (object)` – Объект диска `esu.Disk`

`detach_disk(disk)`

Отсоединить диск от виртуального сервера

Параметры

`disk (object)` – Объект диска `esu.Disk`

`add_port(port)`

Добавить подключение

Параметры

`port (object)` – Новый объект `esu.Port`

`remove_port(port)`

Удалить подключение

Параметры

`port (object)` – Существующий объект *esu.Port*

`power_on()`

Включить виртуальный сервер

`power_off()`

Выключить виртуальный сервер

`reboot()`

Перезагрузить виртуальный сервер

`get_vnc_url()`

Получить ссылку на VNC для открытия консоли управления сервером

Результат

Адрес VNC консоли

Тип результата

str

`revert(snapshot)`

Восстановить сервер из снимка

Параметры

`snapshot (object)` – объект снимка *esu.Snapshot*

`mount_iso(image)`

Примонтировать iso к серверу как CD-ROM. После перезагрузки сервера он будет загружен с этого диска если он загрузочный

Параметры

`image (object)` – объект образа *esu.Image*

`unmount_iso()`

Отмонтировать iso от сервера После перезагрузки сервера он будет загружен с основного диска

2.6.2 Примеры использования

Создать виртуальный сервер на базе ОС Ubuntu 18:

```
from requests import HTTPError
from esu import Manager, VmMetadata, Port, Disk, Vm

vdc = Manager().get_all_vdcs()[0] # Первый доступный пользователю ВЦОД
network = next(n for n in vdc.get_networks() if n.is_default) # Сеть по умолчанию
template = next(v for v in vdc.get_templates() if 'Ubuntu 18' in v.name) # Шаблон ОС
storage_profile = vdc.get_storage_profiles()[0] # Первый доступный профиль хранения
firewall_template = next(f for f in vdc.get_firewall_templates() if f.name == 'По-
↳умолчанию') # Разрешить исходящие подключения
password = 'nw9fH4n$11' # Пароль для виртуального сервера
```

(continues on next page)

(продолжение с предыдущей страницы)

```

metadata = []
for field in template.get_fields():
    value = field.default
    if field.system_alias == 'password':
        value = password
    metadata.append(VmMetadata(field=field, value=value))

port = Port(network=network, fw_templates=[firewall_template])
disk = Disk(name='Системный диск', size=15, storage_profile=storage_profile)

vm = Vm(name='Новый сервер', cpu=2, ram=2, vdc=vdc, template=template,
        metadata=metadata, ports=[port], disks=[disk])

try:
    vm.create()
except HTTPError as ex:
    api_answer = ex.response.json()
    print(f'Error has happend: {api_answer}')

```

Назначить случайный плавающий IP адрес на существующий виртуальный сервер:

```

from esu import Vm, Port

vm = Vm.get_object('954fd467-fd9a-4ce7-b4df-1e81e557bce9')
vm.floating = Port()
vm.save()

```

2.7 Template

Шаблон (операционной) системы. Используется при создании виртуальных серверов. Для получения всех доступных во ВЦОДе шаблонов используется метод `esu.Vdc.get_templates()`.

2.7.1 Объект «шаблон системы»

```
class esu.Template(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор шаблона
- `name (str)` – Имя шаблона
- `min_cpu (int)` – Минимальное количество ядер, необходимое для развертывания этого шаблона
- `min_ram (int)` – Минимальное количество RAM, необходимое для развертывания этого шаблона
- `min_hdd (int)` – Минимальный размер первого диска, необходимого для развертывания этого шаблона

Предупреждение: Объект доступен только для чтения и не может быть создан, изменен или удален.

```
classmethod get_object(id, token=None)
```

Получить объект шаблона по его ID

Параметры

- `id (str)` – Идентификатор шаблона
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект шаблона `esu.Template`

Тип результата

object

```
get_fields()
```

Получить список полей шаблона ОС.

Результат

Список объектов `esu.TemplateField`

Тип результата

list

2.7.2 Примеры использования

См. примеры [здесь](#) и [здесь](#)

2.8 TemplateField

Каждый шаблон системы `esu.Template` содержит список полей шаблона. Он может быть получен с помощью метода `esu.Template.get_fields()`.

Поля специфичны для каждого из шаблонов и являются входными данными, необходимыми для его успешного развертывания в виртуальный сервер.

2.8.1 Объект «поле шаблона»

```
class esu.TemplateField(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор поля шаблона
- `name (str)` – Имя поля шаблона
- `default (str)` – Значение по умолчанию
- `type (str)` – Тип
- `required (boolean)` – Обязательное

- `position (int)` – Порядок
- `system_alias (str)` – Системный идентификатор

Предупреждение: Объект доступен только для чтения и не может быть создан, изменен или удален.

2.8.2 Примеры использования

См. пример [здесь](#)

2.9 VmMetadata

Объект используется при создании и редактировании виртуальных серверов `esu.Vm`

Объект не может быть создан или удален как самостоятельная сущность. Следует управлять списком объектов через свойство класса `metadata` у объекта `esu.Vm`.

2.9.1 Объект «поле метадаты»

```
class esu.VmMetadata(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор
- `field (object)` – Объект `esu.TemplateField`
- `value (str)` – Значение

2.9.2 Примеры использования

См. пример [здесь](#)

2.10 Kubernetes

Кластер Kubernetes. Kubernetes — открытое программное обеспечение для оркестрации контейнеризированных приложений — автоматизации их развёртывания, масштабирования и координации в условиях кластера. Кластер состоит из нескольких серверов - нод. Существует мастер-нода, главная управляющая нода кластера и обычные ноды, которые находятся под управлением мастер-ноды. Для создания кластера необходимо создать или использовать созданный публичный ключ.

2.10.1 Объект «кластер kubernetes»

```
class esu.Kubernetes(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор
- `name (str)` – Имя
- `node_cpu (int)` – CPU нод
- `node_ram (int)` – RAM нод
- `node_disk_size (int)` – Размер диска нод
- `node_storage_profile (object)` – Объект `esu.StorageProfile`
- `nodes_count (int)` – Количество нод в кластере
- `vdc (object)` – Объект класса `esu.Vdc`. ВЦОД, к которому относится данный кластер
- `template (str)` – Идентификатор шаблона Kubernetes
- `user_public_key (string)` – публичный SSH ключ
- `floating (object)` – Объект класса `esu.Port`. Порт подключения кластера к внешней сети. Если `None`, кластер не имеет подключения к внешней сети.

```
classmethod get_object(id, token=None)
```

Получить объект kubernetes по его ID

Параметры

- `id (str)` – Идентификатор Kubernetes
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект кластера kubernetes `esu.Kubernetes`

Тип результата

object

```
create()
```

Создать объект

Исключение

- `ObjectAlreadyHasId` – Если производится попытка создать объект,
- который уже существует –

```
save()
```

Сохранить изменения

```
destroy()
```

Удалить объект

Исключение

`ObjectHasNoId` – Когда производится попытка удалить несуществующий объект

```
get_dashbord_url()
```

Получить ссылку на Dashboard для открытия консоли k8s

Результат

Адрес дашборда

Тип результата

str

2.10.2 Примеры использования

Создать кластер kubernetes:

```
from esu import Vdc, Kubernetes, PublicKey
vdc = Vdc.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
key = PublicKey.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
storage_profile = vdc.get_storage_profiles()[0] # Первый доступный профиль хранения
k8s_template = next(v for v in vdc.get_k8s_templates() if 'Kubernetes 1.22.1' in v.name)
↳ # Шаблон кластера
cluster = Kubernetes(name="MyK8S", vdc=vdc, node_cpu=2, node_ram=2,
                    node_disk_size=10, node_storage_profile=storage_profile,
                    nodes_count=2, template=k8s_template, user_public_key=key)
```

2.11 KubernetesTemplate

Шаблон кластера Kubernetes - необходим для создания кластера Kubernetes. Список шаблонов кластера Kubernetes доступных для созданного ВЦОД можно получить используя метод `esu.Vdc.get_k8s_templates()`

2.11.1 Объект «шаблон кластера kubernetes»

```
class esu.KubernetesTemplate(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор
- `name (str)` – Имя
- `min_node_cpu (int)` – Минимальное CPU нод
- `min_node_ram (int)` – Минимальное RAM нод
- `min_node_hdd (int)` – Минимальный размер диска нод
- `master_template_name (string)` – Название шаблона мастер ноды
- `node_template_name (string)` – Название шаблона ноды
- `vm_node_visible_template (object)` – Объект `esu.Template`
- `hypervisor_type (string)` – Тип ресурсного пула (kvm, vmware)

```
classmethod get_object(id, token=None)
```

Получить объект шаблона по его ID

Параметры

- `id (str)` – Идентификатор шаблона
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект шаблона `esu.KubernetesTemplate`

Тип результата

object

2.11.2 Примеры использования

Запросить список шаблонов кластера kubernetes во ВЦОД:

```
from esu import Vdc

vdc = Vdc.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
k8s_templates = vdc.get_k8s_templates()
```

2.12 Disk

Диск является сущностью для хранения информации. Диск нельзя создать отдельно от виртуального сервера. Чтобы создать диск, следует использовать метод `esu.Vm.add_disk()` у уже существующего сервера.

В то же время, уже созданный диск может быть от него отключен и подключен позднее к другому виртуальному серверу или удален.

Можно изменять размер существующего диска в сторону увеличения.

2.12.1 Объект «диск»

```
class esu.Disk(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор диска
- `name (str)` – Имя диска
- `size (int)` – Размер диска (ГБ)
- `scsi (str)` – Порт, к которому подключен диск
- `vm (object)` – Объект виртуального сервера `esu.Vm`
- `storage_profile (object)` – Объект `esu.StorageProfile`

Примечание: Поля `name`, `size`, `storage_profile` могут быть изменены для существующего объекта.

Предупреждение: `storage_profile` можно изменить только для дисков в сегменте VMware когда диск подключен к виртуальному серверу.

`classmethod get_object(id, token=None)`

Получить объект диска по его ID

Параметры

- `id (str)` – Идентификатор диска
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект диска `esu.Disk`

Тип результата

`object`

`create()`

Создать объект

Исключение

`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует

`save()`

Сохранить изменения

Исключение

`ObjectHasNoId` – Если производится попытка сохранить несуществующий объект

`attach_disk(vm)`

Присоединить существующий во ВЦОДе диск к виртуальному серверу

`detach_disk()`

Отсоединить диск от виртуального сервера

`destroy()`

Удалить объект

Исключение

`ObjectHasNoId` – Когда производится попытка удалить несуществующий объект

2.12.2 Примеры использования

Добавить диск типа SATA к уже существующему виртуальному серверу:

```
from esu import Vm, Disk

vm = Vm.get_object('954fd467-fd9a-4ce7-b4df-1e81e557bce9')

storage_profile = next(p for p in vm.vdc.get_storage_profiles() \
    if p.name == 'SATA')
disk = Disk(name='Дополнительный диск', size=30,
    storage_profile=storage_profile)

vm.add_disk(disk)
```

Увеличить диск виртуального сервера:

```
from esu import Vm

vm = Vm.get_object('954fd467-fd9a-4ce7-b4df-1e81e557bce9')
disk = vm.disks[0]
disk.size += 5
disk.save()
```

2.13 StorageProfile

Объект профиля хранения. Используется при создании дисков виртуальных серверов. Метод `esu.Vdc.get_storage_profiles()` позволяет получить все профили хранения, доступные в определенном ВЦ-Оде.

2.13.1 Объект «профиль хранения»

```
class esu.StorageProfile(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор профиля хранения
- `name (str)` – Имя профиля хранения

Предупреждение: Объект доступен только для чтения и не может быть создан, изменен или удален.

```
classmethod get_object(id, token=None)
```

Получить объект профиля хранения по его ID

Параметры

- `id (str)` – Идентификатор профиля хранения
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект профиля хранения *esu.StorageProfile*

Тип результата

object

2.13.2 Примеры использования

См. пример [здесь](#)

2.14 S3

Хранилище S3. Объектное хранилище используется для хранения файлов. Обладает высокой скоростью обработки запросов и данных. Хранилище создаётся в проекте

2.14.1 Объект «хранилище S3»

```
class esu.S3(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор
- `name (str)` – Имя
- `project (object)` – Объект класса *esu.Project*. Проект, к которому относится данное хранилище
- `access_key (str)` – Ключ доступа к хранилищу
- `secret_key (str)` – Секретный ключ доступа к хранилищу
- `client_endpoint (str)` – URL для подключения к хранилищу

```
classmethod get_object(id, token=None)
```

Получить объект хранилища S3 по его ID

Параметры

- `id (str)` – Идентификатор хранилища S3
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения **ESU_API_TOKEN**

Результат

Возвращает объект хранилища S3 *esu.S3*

Тип результата

object

```
create()
```

Создать объект

Исключение

`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует

save()

Сохранить изменения

destroy()

Удалить объект

Исключение

ObjectHasNoId – Когда производится попытка удалить несуществующий объект

renew_keys()

Сгенерировать новые ключи

2.14.2 Примеры использования

Создать объектное хранилище S3:

```
from esu import S3, Project

project = Project.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
s3 = S3(name="MyS3", project=project)
s3.create()
```

2.15 S3Bucket

Бакет хранилища S3. Файловая корзина - элемент S3 хранилища, который использует для хранения файлов и папок в хранилище S3.

2.15.1 Объект «бакет хранилища S3»

```
class esu.S3Bucket(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- id (*str*) – Идентификатор
- name (*str*) – Имя бакета в ЕСУ
- external_name (*str*) – Имя бакета в S3

```
classmethod get_object(id, s3, token=None)
```

Получить объект бакета хранилища S3 по его ID

Параметры

- id (*str*) – Идентификатор бакета хранилища S3
- token (*str*) – Токен для доступа к API. Если не передан, будет использована переменная окружения **ESU_API_TOKEN**

Результат

Возвращает объект бакета хранилища S3 esu.S3bucket

Тип результата

object

`create(s3)`

Создать объект

Исключение

`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует

`save(s3)`

Сохранить изменения

`destroy(s3)`

Удалить объект

Исключение

`ObjectHasNoId` – Когда производится попытка удалить несуществующий объект

2.15.2 Примеры использования

Создать бакет объектного хранилища S3:

```
from esu import S3, S3bucket

s3 = S3.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
s3bucket = S3Bucket(name="MyS3bucket", s3=s3)
s3bucket.create()
```

2.16 Backup

Задача резервного копирования - создаётся для сервера или для нескольких серверов. Для создания задачи требуется указать расписание: дни и время выполнения задачи, а также серверы для которых она будет выполняться. В каждой задаче резервного копирования хранятся точки восстановления сервера. Из точек восстановления можно восстановить сервер в состояние сохранённое в точке.

2.16.1 Объект «задача резервного копирования»

```
class esu.Backup(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор задачи резервного копирования
- `name (str)` – Имя задачи резервного копирования
- `size (int)` – Суммарный размер точек восстановления задачи резервного копирования (bytes)
- `vdc (object)` – Объект ВЦОДа `esu.Vdc`
- `retain_cycles (int)` – Глубина хранения задачи резервного копирования
- `time (str)` – Время выполнения задачи по расписанию в UTC
- `week_days (list)` – Дни недели выполнения задачи по расписанию [1,2,3]
- `vms (list)` – Список серверов для которых создана задача

Примечание: Поля `name`, `vms`, `retain_cycles`, `week_days`, `time` могут быть изменены для существующего объекта.

`classmethod get_object(id, token=None)`

Получить объект задачи резервного копирования по ее ID

Параметры

- `id (str)` – Идентификатор задачи резервного копирования
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект диска `esu.Disk`

Тип результата

`object`

`create()`

Создать объект

Исключение

`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует

`save()`

Сохранить изменения

Исключение

`ObjectHasNoId` – Если производится попытка сохранить несуществующий объект

`start_immediately()`

Запустить выполнение задачи - создание точек восстановления

`get_restore_points()`

Получить список точек восстановления в задаче резервного копирования.

Результат

Список объектов `esu.RestorePoint`

Тип результата

`list`

`restore(restore_point, power_on=True, quick_restore=False)`

Восстановить сервер из точки восстановления

Параметры

- `vm (object)` – `esu.Vm`, сервер, который необходимо восстановить
- `restore_point (object)` – `esu.RestorePoint`, точка восстановления из которой необходимо восстановить сервер
- `power_on (bool)` – быть включен
- `quick_restore` – (не рекомендуется)

```
get_backup_log()
```

Получить лог создания точек восстановления из задачи

Результат

Отчёт создания точек восстановления

Тип результата

dict

```
get_restore_log(vm)
```

Получить лог восстановления сервера из задачи

Результат

Отчёт восстановления сервера

Тип результата

dict

```
destroy()
```

Удалить объект

Исключение

ObjectHasNoId – Когда производится попытка удалить несуществующий объект

2.16.2 Примеры использования

Создать задачу резервного копирования для сервера:

```
from esu import Vm, Backup, Vdc

vdc = Vdc.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
vm = Vm.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
backup = Backup(name="Test_Backup", vdc=vdc, vms=[vm], week_days=[1, 2],
                time="09:00:00", retain_cycles=2)
```

Немедленно запустить выполнение задачи - создать точку восстановления:

```
import requests
from esu import Backup

backup = Backup.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
backup.start_immediately()
```

Восстановить сервер из точки восстановления:

```
from esu import Vm, Backup

backup = Backup.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
restore_point = backup.get_restore_points()[0] # Первая доступная точка восстановления
backup.restore(restore_point=restore_point)
```

2.17 Image

Образ - образ сервера или iso образ какой либо программы или ОС. Создать образ можно из уже созданного сервера, например для последующего создания копии этого сервера, путем создания сервера из образа. Также в сегменте VMware можно загрузить образ в систему, для последующего создания сервера из образа или для монтирования образа к серверу (iso)

2.17.1 Объект «образ»

```
class esu.Image(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор
- `name (str)` – Имя
- `files (list)` – Список объектов класса `esu.File`. Список файлов в образе –
- `size (str)` – Размер образа
- `type (str)` – Тип образа
- `vdc (object)` – Объект класса `esu.Vdc`. ВЦОД, к которому относится образ

```
classmethod get_object(id, token=None)
```

Получить объект образа по его ID

Параметры

- `id (str)` – Идентификатор образа
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект образа `esu.Image`

Тип результата

object

```
create_from_vm(vm)
```

Создать образ из существующего сервера

Исключение

`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует

```
create_for_upload()
```

Создать объект образа для последующей загрузки в него файлов

Исключение

`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует

```
get_upload_link()
```

Получить ссылку для загрузки файлов образа

`commit_upload()`

Подтвердить окончание загрузки файлов образа. Подтверждение необходимо после загрузки файлов образа по полученному url для загрузки файлов

`save()`

Сохранить изменения

`destroy()`

Удалить объект образа

Исключение

`ObjectHasNoId` – Когда производится попытка удалить несуществующий объект

`get_download_link(file)`

Получить ссылку для скачивания файла образа

`deploy_vm_from_image(vm)`

Создать сервер из образа

Параметры

`vm (object)` –

Результат

объект созданного из образа сервера `esu.Vm`

Тип результата

`object`

2.17.2 Примеры использования

Создать образ из созданного сервера:

```
from esu import Vm, Image, Vdc

vm = Vm.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
vdc = Vdc.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
image = Image(vdc=vdc, name='NewImage')
image.create_from_vm(vm=vm)
```

Загрузить образ:

```
import requests
from esu import Image, Vdc

vdc = Vdc.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
image = Image(vdc=vdc, name='NewImage')
image.create_for_upload() # Создание объекта образа для последующей загрузки файлов
url = image.get_upload_link() # Получение ссылки для загрузки файлов

headers = {
    'Accept': '* / *',
    'Accept-Encoding': 'gzip, deflate, br',
    'Authorization': "bearer " + str(BaseAPI.token),
    'Connection': 'keep-alive',
    'Content-Type': 'application/octet-stream',
```

(continues on next page)

(продолжение с предыдущей страницы)

```

        'Accept-Language': 'ru-ru'
    }
    file = {'file': open('test.iso', 'rb')}
    up_response = requests.put(url=str(url), headers=headers, files=file) # Загрузка файла
    image.commit_upload() # Подтверждение окончания загрузки файлов образа

```

Создание сервера из образа:

```

from esu import Image

image = Image.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
vm = image.deploy_vm_from_image()

```

2.18 Snapshot

Объект снапшота. Снапшот - слепок состояния системы, который можно использовать для последующего восстановления сервера в сохранённое в снапшоте состояние

2.18.1 Объект «снапшот»

```
class esu.Snapshot(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор снапшота
- `name (str)` – Имя снапшота
- `description (str)` – описание для снапшота
- `vm (object)` – Объект класса `esu.Vm`. Сервер, к которому относится данный снапшот

Примечание: Поле `name` и `vm` необходимо для создания

Поля `description` опциональны при создании

Поля `name` и `description` могут быть изменены для существующего объекта

```
classmethod get_object(id, token=None)
```

Получить объект порта по его ID

Параметры

- `id (str)` – Идентификатор снапшота
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект порта `esu.Port`

Тип результата

object

`create()`

Создать объект

Исключение`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует`save()`

Сохранить изменения

Исключение`ObjectHasNoId` – Если производится попытка сохранить несуществующий объект`destroy()`

Удалить объект

Исключение`ObjectHasNoId` – Когда производится попытка удалить несуществующий объект

2.18.2 Примеры использования

Создать снапшот для определённого сервера:

```
from esu import Vm, Snapshot

vm = Vm.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
snapshot = Snapshot(vm=vm, name='Новый снапшот')
snapshot.create()
```

Восстановить сервер из снапшота:

```
from esu import Vm, Snapshot

snapshot = Snapshot.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
vm = Vm.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
vm.revert(snapshot=snapshot)
```

2.19 Network

Приватная сеть обеспечивает обмен данными между различными вычислительными устройствами во ВЦОДе. К примеру, виртуальные машины и маршрутизаторы соединяются сетью. Для правильного функционирования сети, необходимо создать как минимум одну подсеть `esu.Subnet` внутри.

При создании `esu.Vdc` автоматически создается сеть по умолчанию, которую нельзя удалить.

2.19.1 Объект «сеть»

```
class esu.Network(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор сети
- `name (str)` – Имя сети
- `vdc (object)` – Объект класса `esu.Vdc`. ВЦОД, к которому относится сеть
- `is_default (bool)` – True для сети по умолчанию
- `subnets (object)` – Список объектов класса `esu.Subnet`

Примечание: Поля `name` и `vdc` необходимы для создания.

Поле `subnets` опционально при создании.

Поле `name` может быть изменено для существующего объекта.

```
classmethod get_object(id, token=None)
```

Получить объект сети по его ID

Параметры

- `id (str)` – Идентификатор сети
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект сети `esu.Network`

Тип результата

`object`

```
create()
```

Создать объект

Исключение

`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует

```
save()
```

Сохранить изменения

Исключение

`ObjectHasNoId` – Если производится попытка сохранить несуществующий объект

```
destroy()
```

Удалить объект

Исключение

`ObjectHasNoId` – Когда производится попытка удалить несуществующий объект

```
add_subnet(subnet)
```

Добавить подсеть

Параметры

`subnet (object)` – Объект подсети `esu.Subnet`

```
remove_subnet(subnet)
```

Удалить подсеть

Параметры

`subnet` (*object*) – Объект подсети *esu.Subnet*

2.19.2 Примеры использования

Переименование сети по-умолчанию:

```
from esu import Network, Subnet

vdc = Vdc.get_object('e5d9a192-c5da-485a-b134-1b14ec9c57d9')
network = next(n for n in vdc.get_networks() if n.is_default)
network.name = 'Главная сеть'
network.save()
```

2.20 Subnet

Подсеть позволяет задать диапазоны IP-адресов, доступные для использования устройствами, подключенными к определенной сети.

Подсеть не может быть создана или удалена как самостоятельная сущность. Следует использовать методы *esu.Network.add_subnet()* и *esu.Network.remove_subnet()* у уже существующей сети.

2.20.1 Объект «подсеть»

```
class esu.Subnet(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id` (*str*) – Идентификатор
- `cidr` (*str*) – CIDR
- `gateway` (*str*) – Адрес шлюза
- `start_ip` (*str*) – Начальный адрес для DHCP
- `end_ip` (*str*) – Конечный адрес для DHCP
- `enable_dhcp` (*bool*) – Включить или выключить DHCP

2.20.2 Примеры использования

Создание сети с Subnet:

```
from esu import Network, Subnet

subnet = Subnet(cidr='10.22.23.0/24', gateway='10.22.23.1',
                start_ip='10.22.23.2', end_ip='10.22.23.254',
                enable_dhcp=True)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
network = Network(vdc=vdc, name='Network 1', subnets=[subnet])
network.create()
```

Добавление Subnet к уже существующей сети:

```
from esu import Network, Subnet

network = Network.get_object('b9e6df93-0d04-4dac-a3c1-1a8539b8e445')
subnet = Subnet(cidr='10.22.23.0/24', gateway='10.22.23.1',
                start_ip='10.22.23.2', end_ip='10.22.23.254',
                enable_dhcp=True)

network.add_subnet(subnet)
```

2.21 Router

Маршрутизаторы управляют связностью частных сетей с интернетом. Маршрутизатор может иметь собственный публичный адрес, тогда все виртуальные машины будут выходить в Интернет через этот IP-адрес, если на них не назначен собственный плавающий IP.

2.21.1 Объект «маршрутизатор»

```
class esu.Router(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор
- `name (str)` – Имя
- `vdc (object)` – ВЦОД, к которому относится маршрутизатор `esu.Vdc`
- `is_default (bool)` – True для маршрутизатора по умолчанию
- `floating (object)` – Порт подключения к внешней сети `esu.Port`
- `ports (list)` – Список подключений маршрутизатора

Примечание: Поля `name`, `ports` и `vdc` необходимы для создания.

Поле `floating` опционально при создании.

Поля `name` и `floating` могут быть изменены для существующего объекта.

```
classmethod get_object(id, token=None)
```

Получить объект маршрутизатора по его ID

Параметры

- `id (str)` – Идентификатор маршрутизатора
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект маршрутизатора *esu.Router*

Тип результата

object

create()

Создать объект

Исключение

ObjectAlreadyHasId – Если производится попытка создать объект, который уже существует

save()

Сохранить изменения

Исключение

ObjectHasNoId – Если производится попытка сохранить несуществующий объект

destroy()

Удалить объект

Исключение

ObjectHasNoId – Когда производится попытка удалить несуществующий объект

add_port(port)

Добавить подключение

Параметры

port (*object*) – Новый объект *esu.Port*

remove_port(port)

Удалить подключение

Параметры

port (*object*) – Существующий объект *esu.Port*

2.21.2 Примеры использования

Создать новый маршрутизатор и подключить его к первой сети во ВЦОДе:

```
from esu import Vdc, Port, Router

vdc = Vdc.get_object('e5d9a192-c5da-485a-b134-1b14ec9c57d9')
network = vdc.get_networks()[0]
port = Port(network=network)
router = Router(vdc=vdc, name='Новый маршрутизатор', ports=[port])
router.create()
```

Подключить определенный маршрутизатор к существующей сети:

```
from esu import Router, Network, Port

router = Router.get_object('58385696-32c6-4a5c-bafe-895815eedf04')
network = Network.get_object('b9e6df93-0d04-4dac-a3c1-1a8539b8e445')
Port(network=network, device=router).create()
```

2.22 Port

Порт сетевого подключения во ВЦОДе. Используется при подключении виртуальных серверов *esu.Vm* и маршрутизаторов *esu.Router* к сети, а также для подключения публичных IP-адресов к ним.

Если порт используется для подключения виртуального сервера к сети ВЦОД, доступно изменение списка активных шаблонов брандмауэра *esu.FirewallTemplate*.

2.22.1 Объект «порт»

```
class esu.Port(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор порта
- `ip_address (str)` – IP адрес
- `type (str)` – Тип
- `vdc (object)` – Объект класса *esu.Vdc*. ВЦОД, к которому относится данный виртуальный сервер
- `fw_templates (list)` – Включенные шаблоны брандмауэра *esu.FirewallTemplate*
- `network (object)` – Сеть *esu.Network*

Примечание: Поле `network` необходимо для создания в качестве подключения к приватной сети ВЦОД.

Поля `ip_address` и `fw_templates` опциональны при создании подключения к приватной сети ВЦОД

Поля `ip_address` и `fw_templates` могут быть изменены для существующего объекта

При создании подключения плавающего IP обязательных полей нет

```
classmethod get_object(id, token=None)
```

Получить объект порта по его ID

Параметры

- `id (str)` – Идентификатор порта
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект порта *esu.Port*

Тип результата

object

```
create_fip()
```

Получить объект порта по его ID

Параметры

- `id (str)` – Идентификатор порта

- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект порта `esu.Port`

Тип результата

`object`

create()

Создать объект

Исключение

`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует

save()

Сохранить изменения

Исключение

`ObjectHasNoId` – Если производится попытка сохранить несуществующий объект

connect()

Подключить

Исключение

- `ObjectHasNoId` – Если производится попытка присоединить несуществующий объект
- `PortAlreadyConnected` – Если производится попытка присоединить уже присоединенный порт

disconnect()

Отключить порт

Исключение

`ObjectHasNoId` – Если производится попытка сохранить несуществующий объект

destroy()

Удалить объект

Исключение

`ObjectHasNoId` – Когда производится попытка удалить несуществующий объект

force_destroy()

Удалить объект, даже если он подключен к сущности

Исключение

`ObjectHasNoId` – Если производится попытка сохранить несуществующий объект

2.22.2 Примеры использования

См. пример [здесь](#)

2.23 FirewallTemplate

Шаблон брандмауэра. Метод `esu.Vdc.get_firewall_templates()` позволяет получить все шаблоны брандмауэра, доступные в определенном ВЦОДе.

2.23.1 Объект «шаблон брандмауэра»

```
class esu.FirewallTemplate(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор шаблона брандмауэра
- `name (str)` – Имя шаблона брандмауэра
- `vdc (object)` – Объект класса `esu.Vdc`. ВЦОД, к которому относится данный шаблон файрвола
- `description (str)` – описание для шаблона брандмауэра

Предупреждение: Объект доступен только для чтения и не может быть создан, изменен или удален.

```
classmethod get_object(id, token=None)
```

Получить объект шаблона брандмауэра по его ID

Параметры

- `id (str)` – Идентификатор шаблона брандмауэра
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект шаблона брандмауэра `esu.FirewallTemplate`

Тип результата

`object`

```
create()
```

Создать объект

Исключение

`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует

```
save()
```

Сохранить изменения

Исключение

`ObjectHasNoId` – Если производится попытка сохранить несуществующий объект

`destroy()`

Удалить объект

Исключение

`ObjectHasNoId` – Когда производится попытка удалить несуществующий объект

`get_firewall_rules()`

Получить список правил файрвола, доступных в рамках данного шаблона брандмауэра.

Результат

Список объектов `esu.FirewallTemplateRule`

Тип результата

list

2.23.2 Примеры использования

См. пример [здесь](#)

2.24 FirewallTemplateRule

Правило шаблона брандмауэра. Правила добавляются в уже созданный шаблон брандмауэра

2.24.1 Объект «правило шаблона брандмауэра»

```
class esu.FirewallTemplateRule(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор правила брандмауэра
- `name (str)` – Имя правила брандмауэра
- `firewall (str)` – Объект класса `esu.FirewallTemplate`. Шаблон брандмауэра, к которому относится данное правило брандмауэра
- `direction (str)` – направление правила брандмауэра
- `destination_ip (str)` – `destination_ip` правила брандмауэра
- `dst_port_range_max (str)` – `dst_port_range_max` правила брандмауэра
- `dst_port_range_min (str)` – `dst_port_range_min` правила брандмауэра
- `protocol (str)` – `protocol` правила брандмауэра
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Примечание: Поля `direction`, `name` и `protocol` необходимы для создания.

```
classmethod get_object(firewall, rule_id, token=None)
```

Получить объект правил брандмауэра по его ID

Параметры

- `id (str)` – Идентификатор правила шаблона брандмауэра
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект правила шаблона брандмауэра `esu.FirewallTemplateRule`

Тип результата

object

`create()`

Создать объект

Исключение

`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует

`save()`

Сохранить изменения

Исключение

`ObjectHasNoId` – Если производится попытка сохранить несуществующий объект

`destroy()`

Удалить объект

Исключение

`ObjectHasNoId` – Когда производится попытка удалить несуществующий объект

2.24.2 Примеры использования

Создать правило для шаблона брандмауэра:

```
from esu import FirewallTemplate, FirewallTemplateRule

fw_template = FirewallTemplate.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
firewall_rule = QAFirewallRule(firewall=fw_template, name="Rule",
                                direction="egress", protocol="tcp",
                                destination_ip="0.0.0.0/0")

firewall_rule.create
```

2.25 Dns

Доменная зона. В доменной зоне могут быть заведены DNS записи различных типов. Метод `esu.Project.get_dns_zones` позволяет получить все доменные зоны доступные в проекте.

2.25.1 Объект «доменная зона»

```
class esu.Dns(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор Dns
- `name (str)` – Имя Dns
- `project (object)` – Объект класса `esu.Project`. Проект, к которому относится данный Dns
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Примечание: Поля `name` и `project` необходимы для создания.

Поле `name` может быть изменено для существующего объекта.

```
classmethod get_object(id, token=None)
```

Получить объект Dns по его ID

Параметры

- `id (str)` – Идентификатор Dns
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект Dns `esu.Dns`

Тип результата

object

```
create()
```

Создать объект

Исключение

`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует

```
save()
```

Сохранить изменения

Исключение

`ObjectHasNoId` – Если производится попытка сохранить несуществующий объект

```
destroy()
```

Удалить объект

Исключение

`ObjectHasNoId` – Когда производится попытка удалить несуществующий объект

```
get_dns_records()
```

Получить список днс записей, доступных в рамках данного Dns.

Результат

Список объектов `esu.DnsRecord`

Тип результата
list

2.25.2 Примеры использования

Создать доменную зону в проекте:

```
from esu import Project, Dns

project = Project.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
dns = Dns(project=project, name='test.com')
dns.create()
```

2.26 DnsRecord

Запись доменной зоны. В доменной зоне могут быть заведены DNS записи различных типов

2.26.1 Объект «запись доменной зоны»

```
class esu.DnsRecord(*args, token: Optional[str] = None, **kwargs)
```

Параметры

- `id (str)` – Идентификатор Dns записи
- `dns_id (str)` – Объект класса `esu.Dns`. Dns зона, к которому относится dns запись
- `data (str)` – дата Dns записи
- `flag (str)` – флаг Dns записи
- `host (str)` – хост Dns записи
- `port (str)` – порт Dns записи
- `priority (str)` – приоритет Dns записи
- `tag (str)` – тэг Dns записи
- `ttl (str)` – ttl Dns записи
- `type (str)` – тип Dns записи
- `weight (str)` – вес Dns записи
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Примечание: Поля `data`, `dns`, `host`, `ttl`, `type` необходимы для создания.

```
classmethod get_object(dns, id, token=None)
```

Получить объект Dns запись по его ID

Параметры

- `id (str)` – Идентификатор Dns записи
- `token (str)` – Токен для доступа к API. Если не передан, будет использована переменная окружения `ESU_API_TOKEN`

Результат

Возвращает объект Dns `esu.DnsRecord`

Тип результата

object

create()

Создать объект

Исключение

`ObjectAlreadyHasId` – Если производится попытка создать объект, который уже существует

save()

Сохранить изменения

Исключение

`ObjectHasNoId` – Если производится попытка сохранить несуществующий объект

destroy()

Удалить объект

Исключение

`ObjectHasNoId` – Когда производится попытка удалить несуществующий объект

2.26.2 Примеры использования

Создать запись доменной зоны в доменной зоне:

```
from esu import Dns, DnsRecord

dns = Dns.get_object('d5cd2cdc-b5b0-4d2e-8bc6-ea3f019745f9')
record = DnsRecord(type="A", host="host", data="10.0.1.1",
                   dns=dns, ttl=86400)
record.create()
```

2.27 0.1.6

Обновление от 2021-07-07

- При создании ВЦОДа вместо параметра `hypervisor_type` следует передавать параметр `hypervisor`, являющийся объектом `esu.Hypervisor`. Получить список всех доступных гипервизоров можно, вызвав метод `esu.Project.get_available_hypervisors()`

A

add_disk() (метод *esu.Vm*), 15
 add_port() (метод *esu.Router*), 37
 add_port() (метод *esu.Vm*), 15
 add_subnet() (метод *esu.Network*), 34
 attach_disk() (метод *esu.Disk*), 23
 attach_disk() (метод *esu.Vm*), 15

B

Backup (класс в *esu*), 27

C

Client (класс в *esu*), 7
 commit_upload() (метод *esu.Image*), 30
 connect() (метод *esu.Port*), 39
 create() (метод *esu.Backup*), 28
 create() (метод *esu.Disk*), 23
 create() (метод *esu.Dns*), 43
 create() (метод *esu.DnsRecord*), 45
 create() (метод *esu.FirewallTemplate*), 40
 create() (метод *esu.FirewallTemplateRule*), 42
 create() (метод *esu.Kubernetes*), 20
 create() (метод *esu.Network*), 34
 create() (метод *esu.Port*), 39
 create() (метод *esu.Project*), 8
 create() (метод *esu.Router*), 37
 create() (метод *esu.S3*), 25
 create() (метод *esu.S3Bucket*), 26
 create() (метод *esu.Snapshot*), 33
 create() (метод *esu.Vdc*), 11
 create() (метод *esu.Vm*), 15
 create_fip() (метод *esu.Port*), 38
 create_for_upload() (метод *esu.Image*), 30
 create_from_vm() (метод *esu.Image*), 30
 create_vm() (метод *esu.Vdc*), 13

D

deploy_vm_from_image() (метод *esu.Image*), 31
 destroy() (метод *esu.Backup*), 29

destroy() (метод *esu.Disk*), 23
 destroy() (метод *esu.Dns*), 43
 destroy() (метод *esu.DnsRecord*), 45
 destroy() (метод *esu.FirewallTemplate*), 40
 destroy() (метод *esu.FirewallTemplateRule*), 42
 destroy() (метод *esu.Image*), 31
 destroy() (метод *esu.Kubernetes*), 20
 destroy() (метод *esu.Network*), 34
 destroy() (метод *esu.Port*), 39
 destroy() (метод *esu.Project*), 9
 destroy() (метод *esu.Router*), 37
 destroy() (метод *esu.S3*), 26
 destroy() (метод *esu.S3Bucket*), 27
 destroy() (метод *esu.Snapshot*), 33
 destroy() (метод *esu.Vdc*), 11
 destroy() (метод *esu.Vm*), 15
 detach_disk() (метод *esu.Disk*), 23
 detach_disk() (метод *esu.Vm*), 15
 disconnect() (метод *esu.Port*), 39
 Disk (класс в *esu*), 22
 Dns (класс в *esu*), 43
 DnsRecord (класс в *esu*), 44

F

FirewallTemplate (класс в *esu*), 40
 FirewallTemplateRule (класс в *esu*), 41
 force_destroy() (метод *esu.Port*), 39

G

get_all_clients() (метод *esu.Manager*), 5
 get_all_firewall_templates() (метод *esu.Manager*), 6
 get_all_networks() (метод *esu.Manager*), 6
 get_all_platforms() (метод *esu.Manager*), 6
 get_all_projects() (метод *esu.Manager*), 5
 get_all_storage_profiles() (метод *esu.Manager*), 6
 get_all_vdcs() (метод *esu.Manager*), 6
 get_all_vms() (метод *esu.Manager*), 6

- get_available_hypervisors() (метод *esu.Project*), 9
 get_backup_log() (метод *esu.Backup*), 28
 get_backups() (метод *esu.Vdc*), 13
 get_dashbord_url() (метод *esu.Kubernetes*), 20
 get_disks() (метод *esu.Vdc*), 12
 get_dns_records() (метод *esu.Dns*), 43
 get_dns_zones() (метод *esu.Project*), 9
 get_download_link() (метод *esu.Image*), 31
 get_fields() (метод *esu.Template*), 18
 get_firewall_rules() (метод *esu.FirewallTemplate*), 41
 get_firewall_templates() (метод *esu.Vdc*), 12
 get_images() (метод *esu.Vdc*), 13
 get_k8s_clusters() (метод *esu.Project*), 9
 get_k8s_templates() (метод *esu.Vdc*), 13
 get_networks() (метод *esu.Vdc*), 12
 get_object() (метод класса *esu.Backup*), 28
 get_object() (метод класса *esu.Client*), 7
 get_object() (метод класса *esu.Disk*), 23
 get_object() (метод класса *esu.Dns*), 43
 get_object() (метод класса *esu.DnsRecord*), 44
 get_object() (метод класса *esu.FirewallTemplate*), 40
 get_object() (метод класса *esu.FirewallTemplateRule*), 41
 get_object() (метод класса *esu.Image*), 30
 get_object() (метод класса *esu.Kubernetes*), 20
 get_object() (метод класса *esu.KubernetesTemplate*), 21
 get_object() (метод класса *esu.Network*), 34
 get_object() (метод класса *esu.Port*), 38
 get_object() (метод класса *esu.Project*), 8
 get_object() (метод класса *esu.Router*), 36
 get_object() (метод класса *esu.S3*), 25
 get_object() (метод класса *esu.S3Bucket*), 26
 get_object() (метод класса *esu.Snapshot*), 32
 get_object() (метод класса *esu.StorageProfile*), 24
 get_object() (метод класса *esu.Template*), 18
 get_object() (метод класса *esu.Vdc*), 11
 get_object() (метод класса *esu.Vm*), 15
 get_ports() (метод *esu.Vdc*), 12
 get_projects() (метод *esu.Client*), 7
 get_restore_log() (метод *esu.Backup*), 29
 get_restore_points() (метод *esu.Backup*), 28
 get_routers() (метод *esu.Vdc*), 12
 get_s3_storages() (метод *esu.Project*), 9
 get_storage_profiles() (метод *esu.Vdc*), 12
 get_templates() (метод *esu.Vdc*), 12
 get_upload_link() (метод *esu.Image*), 30
 get_vdcs() (метод *esu.Project*), 9
 get_vms() (метод *esu.Vdc*), 11
 get_vnc_url() (метод *esu.Vm*), 16
- ## H
- Hypervisor (класс в *esu*), 10
- ## I
- Image (класс в *esu*), 30
- ## K
- Kubernetes (класс в *esu*), 20
 KubernetesTemplate (класс в *esu*), 21
- ## M
- Manager (класс в *esu*), 5
 mount_iso() (метод *esu.Vm*), 16
- ## N
- Network (класс в *esu*), 34
- ## P
- Port (класс в *esu*), 38
 power_off() (метод *esu.Vm*), 16
 power_on() (метод *esu.Vm*), 16
 Project (класс в *esu*), 8
- ## R
- reboot() (метод *esu.Vm*), 16
 remove_port() (метод *esu.Router*), 37
 remove_port() (метод *esu.Vm*), 16
 remove_subnet() (метод *esu.Network*), 34
 renew_keys() (метод *esu.S3*), 26
 restore() (метод *esu.Backup*), 28
 revert() (метод *esu.Vm*), 16
 Router (класс в *esu*), 36
- ## S
- S3 (класс в *esu*), 25
 S3Bucket (класс в *esu*), 26
 save() (метод *esu.Backup*), 28
 save() (метод *esu.Disk*), 23
 save() (метод *esu.Dns*), 43
 save() (метод *esu.DnsRecord*), 45
 save() (метод *esu.FirewallTemplate*), 40
 save() (метод *esu.FirewallTemplateRule*), 42
 save() (метод *esu.Image*), 31
 save() (метод *esu.Kubernetes*), 20
 save() (метод *esu.Network*), 34
 save() (метод *esu.Port*), 39
 save() (метод *esu.Project*), 9
 save() (метод *esu.Router*), 37
 save() (метод *esu.S3*), 25
 save() (метод *esu.S3Bucket*), 27
 save() (метод *esu.Snapshot*), 33
 save() (метод *esu.Vdc*), 11
 save() (метод *esu.Vm*), 15

Snapshot (класс в esu), 32
start_immediately() (метод esu.Backup), 28
StorageProfile (класс в esu), 24
Subnet (класс в esu), 35

T

Template (класс в esu), 17
TemplateField (класс в esu), 18

U

unmount_iso() (метод esu.Vm), 16

V

Vdc (класс в esu), 11
Vm (класс в esu), 14
VmMetadata (класс в esu), 19