rustack-esu

Development Team

сент. 07, 2021

Управление ресурсами

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.

Глава 1

Установка

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

Глава 2

Начало работы

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 сотронним 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

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 сиок 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

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 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**.

2.4.1 Объект «ВЦОД»

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

Параметры

- `id` (*str*) – Идентификатор ВЦОД
- `name` (*str*) – Имя ВЦОД
- `hypervisor_type` (*str*) – Тип гипервизора. **vmware** или **kvm**
- `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_type` 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.

Результат Список объектов *esu.Vm*

Тип результата list

`get_templates()`
Получить список шаблонов ОС для создания виртуальных машин, доступных в данном ВЦОДе.

Результат Список объектов *esu.Template*

Тип результата list

`get_storage_profiles()`
Получить список профилей хранения, которые используются при добавлении дисков, доступных в данном ВЦОДе.

Результат Список объектов *esu.StorageProfile*

Тип результата list

`get_firewall_templates()`
Получить список шаблонов брандмауэра, доступных в данном ВЦОДе.

Результат Список объектов *esu.FirewallTemplate*

Тип результата list

`get_networks()`
Получить список сетей, которые существуют в рамках данного ВЦОД.

Результат Список объектов *esu.Network*

Тип результата list

`get_routers()`
Получить список маршрутизаторов, которые доступны в рамках данного ВЦОД.

Результат Список объектов *esu.Router*

Тип результата list

`get_ports()`
Получить список подключений, которые существуют в данном ВЦОДе.

Результат Список 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

2.4.2 Примеры использования

Создание нового ВЦОД VMware в первом доступном пользователю проекте:

```
from esu import Manager, Vdc

project = Manager().get_all_projects()[0]
vdc = Vdc(name='Новый ВЦОД', hypervisor_type='vmware', 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)
```

2.5 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.

2.5.1 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 виртаульаного 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

`power_on()`
Включить виртуальный сервер

`power_off()`
Выключить виртуальный сервер

`reboot()`
Перезагрузить виртуальный сервер

`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*

`get_vnc_url()`
Получить ссылку на VNC для открытия консоли управления сервером

Результат Адрес VNC консоли

Тип результата str

2.5.2 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

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()
```

2.6 Template

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

2.6.1 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

2.6.2 Примеры использования

См. примеры *здесь* и *здесь*

2.7 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.

2.7.1 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.

2.7.2 Примеры использования

См. пример *здесь*

2.8 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*.

2.8.1 Объект «поле метадаты»

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

Параметры

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

2.8.2 Примеры использования

См. пример *здесь*

2.9 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.

2.9.1 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

save()
Сохранить изменения

Исключение ObjectHasNoId – Если производится попытка сохранить несуществующий объект

destroy()
Удалить объект

Исключение ObjectHasNoId – Когда производится попытка удалить несуществующий объект

2.9.2 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()
```

2.10 StorageProfile

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

2.10.1 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

2.10.2 Примеры использования

См. пример *здесь*

2.11 Network

Приватная сеть обеспечивает обмен данными между различными вычислительными устройствами во ВЦОДе. К примеру, виртуальные машинамы 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.

2.11.1 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*

2.11.2 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()
```

2.12 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.

2.12.1 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

2.12.2 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)

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)
```

2.13 Router

Маршрутизаторы управляют связностью приватных 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.

2.13.1 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*

2.13.2 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')
router.add_port(Port(network=network))
```

2.14 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*.

2.14.1 Объект «порт»

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

Параметры

- id (*str*) – Идентификатор порта
- ip_address (*str*) – IP адрес
- type (*str*) – Тип
- 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

save()

Сохранить изменения

Исключение ObjectHasNoId – Если производится попытка сохранить несуществующий объект

2.14.2 Примеры использования

См. пример *здесь*

2.15 FirewallTemplate

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

2.15.1 Объект «шаблон брандмауэра»

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

Параметры

- id (*str*) – Идентификатор шаблона брандмауэра
- name (*str*) – Имя 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

2.15.2 Примеры использования

См. пример *здесь*

Алфавитный указатель

S

T

V