RackCorp REST API

Настройка фреймворка RackCorp REST API

- REST API Архитектура и стандарты
- RACKCORP REST API
- ПРИМЕР RACKCORP REST API

REST API ??????????? ? ?????????

Архитектура REST API была создана, чтобы упростить и стандартизировать процедуры доступа к данным из различных служб.

Эта логическая архитектура часто использует структуру данных JSON для отправки и получения данных. JSON — это открытый стандартный формат файла и формат обмена данными, в котором для хранения и передачи объектов данных, состоящих из пар атрибут-значение и массивов, используется удобочитаемый текст.

Пример данных JSON:

```
{
"api": "v2.7",
    "apipath": "https://www.rackcorp.net/api/v2.7"
    }
```

Эти данные могут быть представлены в виде массива или объекта на многих языках, таких как PHP, PYTHON и C#.

Rackcorp перенесла многие из своих функций для работы с архитектурой REST API, чтобы модернизировать и упростить процесс получения данных по протоколу HTTP. Каждый месяц мы добавляем новые сервисы через REST API. Важно, чтобы вы подписались на нас, чтобы получать последние обновления и последнюю версию нашего API.

????????? ?????? ????????? REST API:

- **методы HTTP - GET, POST, PUT, DELETE**

  Веб-разработчики, вероятно, знакомы с GET и POST, а также с другими методами HTTP, также иногда называемыми HTTP-глаголами. Эти методы определяют тип запроса к REST API.

- **Методы по использованию:**
  - GET - запросить данные
  - POST - вставить новые данные
  - PUT - обновить данные
  - DELETE - удалить данные

- **Имена ресурсов:**

Resources are sometimes referred to as the nouns that the HTTP verbs act upon. Earlier web services were built around remote procedure calls, which saw APIs as extensions of the code that called them. By contrast, REST resources can be accessed with multiple HTTP methods.

Ресурсы иногда называют существительными, на которые воздействуют HTTP-глаголы. Ранее веб-службы были построены на основе удаленных (на дистанции) вызовов процедур, в которых API рассматривались как расширения вызывающего их кода. Напротив, к ресурсам REST можно получить доступ с помощью нескольких методов HTTP.

- `GET /api/animals:` получить список животных.
- `POST /api/animals:` добавить новое животное
- `GET /api/animals/dog:` получить одно животное по идентификатору ID.
- `PUT /api/animals/dog:` обновить одно животное по идентификатору ID.
- `DELETE /api/animals/dog:` удалить животное по идентификатору ID.

- **Форматы данных:**

  Большинство запросов API будут возвращать с сервера содержимое, которое клиент должен интерпретировать. Редко это содержимое представляет собой обычный текст — обычно оно использует формат структурированных данных. Хотя REST не определяет какие-либо форматы данных, чаще всего используются JSON и XML.

  JSON:

```
{
  "id": "dog",
  "name": "Pet dog",
  "genus": "Canis",
  "img": "https://cdn2.thedogapi.com/images/1MZ0YbOpS.jpg"
}
```

  XML:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <id>dog</id>
  <name>Domestic dog</name>
  <genus>Canis</genus>
  <img>https://cdn2.thedogapi.com/images/1MZ0YbOpS.jpg
</img>
</root>
```

- **Статусы HTTP:**

  Поскольку API-интерфейсы REST зависят от стандартов HTTP, статус каждого запроса используется для сообщения результата запроса, такого как успех или неудача. Каждый код состояния предоставляет машиночитаемый ответ, а также сообщение, понятное человеку. Веб-разработчики (и ряд пользователей) знакомы со многими из них.

  - **200**: Удачный (Success)
  - **201**: Создан (Created)
  - **401**: Неавторизирован (Unauthorized)
  - **403**: Запрещен (Forbidden)
  - **404**: Не найден (Not found)
  - **429**: Слишком много запросов (Too many requests)

Хотя REST не является стандартом, существует множество других стандартов, часто связанных с REST. Например, OAuth охватывает стороннюю авторизацию ресурсов, а JSON PATCH описывает стандартный подход к методу HTTP PATCH для формата данных JSON. Важным стандартом, о котором следует помнить при разработке собственных API, является спецификация OpenAPI.

??? ??????? ?????? REST API ??? ?????????, ??????? ? ?????????? ???????

Вы можете создать REST API, используя любой язык программирования, такой как Javascript, PHP, PYTHON и C#. В этой статье мы добавим несколько примеров кода, чтобы объяснить, как сделать запрос с помощью Javascript (JQUERY) и PHP. Мы выбрали эти два языка, потому что это они самые популярные для развертывания веб-сайтов.

> **Мы настоятельно рекомендуем выполнять вызовы REST API только для бэкэнда. Будьте осторожны при использовании кода javascript для вызова URL-адреса REST API. Никогда не добавляйте свои учетные данные во фронтъенд.**

Javscript (JQUERY):

Во-первых, Вам нужно создать небольшую логику для вызова URL-адреса API в качестве примера ниже:

```
<script>
let query = {};
const v = 'v2.7';
const URL = 'https://www.rackcorp.net/api/'+v+'/dcs';
```

```
$.ajax({
ur: URL,
    type: 'GET',
    data: query,
    success: function(res) {
        console.log(res);
        alert(res);
    },
    error: function(res) {
    console.log(res);
        alert(res);
    }
});
</script>
```

Вы можете видеть, что код для выполнения запросов с использованием REST API довольно прост.

У нас есть переменная с именем «data», которая представляет собой объект, который будет содержать пару «ключ-значение», которая будет преобразована в запрос в бэкэнде. Затем мы создаем строковую переменную, которая является URL-адресом для подключения к REST API HTTP. С помощью этих двух переменных мы можем создать скрипт AJAX для вызова REST API. Если код правильный, логика перенаправит в SUCCESS, если код неправильный, он перенаправит к ERROR. Любой другой вариант Вы можете редактировать по своему усмотрению.

Будьте осторожны, чтобы использовать javascript для выполнения запросов REST API. Это связано с тем, что javascript является интерфейсным (фронт-энд) языком программирования, это означает, что код запускается на рабочем столе пользователя (ноутбук, телефон) и этот код может быть перехвачен хакерами и любым другим злоумышленником. Запросы REST API в javascript рекомендуются только в том случае, если вам не нужно передавать конфиденциальные данные, такие как ключи, пароль или личную информацию. В приведенном выше примере мы вызвали URL-адрес, чтобы перечислить все центры обработки данных (датацентры или DC) в Rackcorp, которые не требуют аутентификации, они открыты для всего мира.

PHP:

В PHP обычно разработчики используют функцию CURL для подключения через HTTP. В примере ниже вы можете понять, как реализовать запрос REST API с вашего сервера на другой сервер.

```
<?php
```

```
$v = 'v2.7';
$url = "https://api.rackcorp.net/api/". $v . "/dcs";
$query = [];

$curl = curl_init($url);
curl_setopt($curl, CURLOPT_POSTFIELDS, json_encode($query));
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);

$response = curl_exec($curl);

if($response) {
return json_decode($response, true);
}

return false;
```

Как видите, мы создали одни и те же переменные «v» и «query» в PHP для вызова URL-адреса REST API. На них мы начинаем создавать начальную команду CURL. Мы устанавливаем некоторые параметры в функции CURL, такие как POSTFIELDS и RETURNTRANSFER. У вас есть много других опций, которые вы можете добавить в CURL PHP. Просмотрите документацию по PHP, чтобы найти наилучший подход к логике вашего кода.

?????:

Запросы REST API — это довольно простой способ получать, вставлять и обновлять данные через Интернет с небольшими усилиями. В настоящее время в большинстве сервисов через Интернет реализованы функции REST API, и RACKCORP также имеет свой собственный стандарт. Если вам нужно подключиться к нам с помощью REST API, узнайте больше в RACKCORP REST API.

RACKCORP REST API

Rackcorp перенесла многие из своих функций для работы с архитектурой REST API, чтобы модернизировать и упростить процесс получения данных по протоколу HTTP. Каждый месяц мы добавляем новые сервисы через REST API. Важно, чтобы Вы подписались на нас, чтобы получать последние обновления и последнюю версию нашего API.

?????????? ?? API:

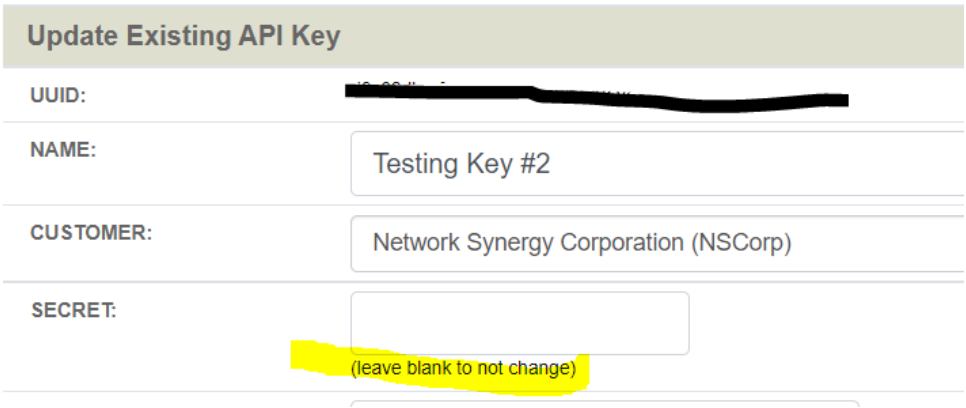
Текущая версия: v2.8

Ссылка на API: https://www.rackcorp.net/api/v2.8

Прежде чем Вы начнете создавать какой-либо код или подключаться через наш API, Вам необходимо создать доступ к API-ключу, который позволит Вашему коду отправлять HTTP-запросы на нашу сторону и авторизоваться для получения данных из Ваших служб. Для этого Вам нужно связаться с нами по форме обратной связи.

Чтобы создать учетные данные API, перейдите в АДМИНИСТРИРОВАНИЕ -> API на нашем портале. URL: https://portal.rackcorp.com/index.php?cmd=api

Затем нажмите ДОБАВИТЬ, введите имя для этого нового ключа и секрет (пароль) и СОХРАНИТЕ



> Обязательно запишите свою СЕКРЕТНУЮ фразу в безопасном месте. Он необходим для доступа к API и не может быть получен. Его можно сбросить только на странице сведений о ключе портала API.

???????? API:

Как описано в нашей статье «Архитектура и стандарты REST API», не существует определенных стандартных шаблонов, которым должны следовать все инженеры/разработчики при создании REST API для своего приложения. В Rackcorp все просто. Основная структура нашего REST API — это URL-адрес, который соответствует приведенным ниже шаблонам:

Версия: v2.8

Главный URL: https://api.rackcorp.net/api

Данные объекта: клиенты | DCS | сеть | API | днс | устройства

Формат URL: https://api.rackcorp.net/api/+version+/object data/+ID item

Пример запроса GET для всех данных: https://api.rackcorp.net/api/v2.8/device

Пример запроса GET на получение одного типа данных:

https://api.rackcorp.net/api/v2.8/device/1

Ниже вы видите код, который Вы можете использовать в качестве примера для реализации Вашего первого подключения REST API к нашей платформе:

> **Мы настоятельно рекомендуем выполнять вызовы REST API только из бэкэнда. Будьте осторожны при использовании кода javascript для вызова URL-адреса REST API. Никогда не добавляйте свои учетные данные во фронтэнд.**

В этом простом введении в код REST API мы будем использовать языки программирования PHP и PYTHON.

PHP:

В примере ниже мы добавляем фиктивные API KEY и API SECRET. Пожалуйста, измените эти данные в соответствии с вашими данными

```
<?php
// Simple example to get a list of all datacenters
$version = 'v2.8';
$url = "https://api.rackcorp.net/api/". $version . "/dc";
$query = ["cmd"=>"dc.getall"];
$query['APIUUID'] = "";  // No authetnication required for getting datacenter list
$query['APISECRET'] = "";
```

```
$curl = curl_init($url);
curl_setopt($curl, CURLOPT_POSTFIELDS, json_encode($query));
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);

$response = curl_exec($curl);

if($response) {
return json_decode($response, true);
}

return false;
?>
```

PYTHON:

```
import json
import logging
import sys
import os
import glob
import re
from bson import json_util
from flask.helpers import make_response
from flask import request, jsonify

version = 'v2.8'
apiurl = 'https://api.rackcorp.net/api'+version+'/'+dc
setheaders = {
        'content-type': 'application/json',
        'User-Agent': 'Mozilla',
        'jwt': jwt
}

data = {}
data['APIUUID'] = ''
data['APISECRET'] = ''

apiresp = None
apiresp = requests.get(self.apiurl+'/sessions/logout',data=self.rcdata,headers=setheaders)
```

Как Вы можете видеть в обоих примерах, логика подключения через наш API довольно проста. Вам просто нужно, чтобы URL-адрес, APIUUID и APISECRET были частью объекта или массива, а также переменные, которые Вы передаете через API для построения запроса в бэкэнде.

Надеюсь, Вы получили общее представление о том, как использовать наши функции REST API. Ниже Вы можете найти ссылки с более продвинутой документацией по использованию нашего API. Кроме того, полный список служб REST API (URL-адреса) с объяснением данных запроса и ожидаемых данных ответа для каждой ситуации.

REST API GibHub Docs:

Link: https://github.com/RackCorpCloud/rackcorp-api/wiki/RACKCORP-REST-API

Swagger RACKCORP REST API:

Swagger — это набор инструментов разработчика API от SmartBear Software и бывшая спецификация, на которой основана спецификация OpenAPI. Эта платформа отображает в простом макете все функции REST API, позволяя Вам визуализировать то, что Ваш код должен ожидать в качестве ответа на каждый вызов (GET, PUT, PUSH, DELETE).

Через эту платформу Вы можете увидеть схему для каждой функции, а также ожидаемый ответ в формате JSON. Вы также можете сделать тесты для подключения через свои сервисы на наших серверах, используя свои настоящие APIUUID и APISECRET.

Это хорошо для тестирования промежуточных сред и убедиться, что Ваш вызов получит именно то, что ожидает Ваш код.

> Не забудьте выбрать API пути URL, который Вы хотите использовать для тестов. На странице Swagger Вы можете увидеть три варианта. Первый — это URL-адрес, который не предназначен для тестов. Второй — RACKCORP Production REST API Core (пожалуйста, будьте осторожны при использовании этого URL-адреса). Третий — RACKCORP Staging REST API, который следует использовать для тестов.

Link: https://app.swaggerhub.com/apis/RackCorp/Rackcorp-REST-API/2.8

?????? RACKCORP REST API

PHP:

Простое создание сервера:

```
<?php
function rackcorpAPI($action, $request) {
        $request["APIUUID"] = "";
        $request["APISECRET"] = "";
        $request["cmd"] = $action;

        $curl = curl_init("https://api.rackcorp.net/api/rest/v2.7/json.php");
        curl_setopt($curl, CURLOPT_POST, true);
        curl_setopt($curl, CURLOPT_POSTFIELDS, json_encode($request));
        curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
        $response = curl_exec($curl);
        curl_close($curl);
        if ( !$response ) {
                return Array("code" => "FAULT", "message" => "API Error");
        }
        return json_decode($response, true);
}

$customerID = 1000;  // Change this to your customer ID (available in portal under
ADMINISTRATION -> MY DETAILS)
// locations are defined here: https://wiki.rackcorp.com/books/help-and-support-
en/page/rackcorp-datacenter-locations-and-codes

$neworder = Array(
    "productCode" => "SERVER_VIRTUAL_PERFORMANCE_AU",
    "customerId" => $customerID,
    "quantity" => 1,
    "productDetails" => Array(
        "hostname" => "testrouter01",
        "cpu" => 4,
        "memoryGB" => 4,
        "trafficGB" => 100,
```

```
        "nics" => Array(
            Array(
               "name" => "public",
               "speed" -> 1000,
               "ipv4" => 1,
            )
        ),
        "storage" => Array(
            Array(
                "name" => "MAIN",
                "sizeGB" => 30,
                "type" => "SSD",
                "order" => 1
            )
        ),
        "credentials" => Array(
            Array(
                "username" => "root",
                "password" => "DUGHsUIYGUI312TEST"
            )
        ),
        "location" => "RC-AU-GLOBESW1",
        "timezone" => "UTC",
        "install" => Array (
            "operatingSystem" => "UBUNTU20.04_64",
            "template" => "standard",
            "postInstallScript" => ""
        )
    )
);
// Lodge the order (this just locks pricing in for up to 72 hours but doesnt actually create
any resources)
$response = rackcorpAPI("order.create", $neworder);
var_dump($response);

// You can look up the order if you want:
$neworder = Array ("orderId" => $response["orderId"]);
$response = rackcorpAPI("order.getall", $neworder);
var_dump($response);
```

```
// Then confirm the order to start provisioning:
$neworder = Array ("orderId" => $response["orderId"]);
$response = rackcorpAPI("order.confirm", $neworder);
var_dump($response);

?>
```

Запуск сервера с помощью cloud-init:

После создания сервера вы также можете запустить его с помощью cloud-init со своим собственным кодом:

```
$cloudInitStartupData = Array(
        "cloudInit" => Array(
                "volumeName" => "config-2",
                "userData" => "#cloud-config
ssh_pwauth: True
users:
  - default
  - name: user1
    groups: sudo
    shell: /bin/bash
    sudo: ['ALL=(ALL) NOPASSWD:ALL']
    plain_text_passwd: testtest888
    lock_passwd: false
",
                "metaData" => "instance-id: ServerTest9999
local-hostname: MyServerHostname9999
"
        )
);
$serverIDToStart = 9999;
$tx = Array ("objId"=>$serverIDToStart, "objType"=>"DEVICE", "type"=>"STARTUP",
"data"=>json_encode($cloudInitStartupData));

// See earlier example for rackcorpAPI function
$response = rackcorpAPI("rctransaction.create", $tx);
var_dump($response);
?>
```