

Table of contents

API Overview	2
API Quickstart	3
API Reference	5
Create user	6
Create list of users with given input array	9
Log user into the system	13
Log current user out	15
Find user	16
Update user	18
Delete user	21

API Overview

Introduction

Provide a brief introduction to the API, explaining its purpose and scope.

What you can do using this API

Provide some simple usage examples to help users get started quickly.

Authentication

Explain the authentication methods and requirements for accessing the API.

Base URL

Specify the base URL for making API requests.

If you have more than one environment (production and sandbox) explain the difference and provide links to both.

Rate Limiting

Explain any rate limiting policies, if applicable.

Error Handling

Describe the API's error response format and provide common error codes and their meanings.

Versioning

Explain how the API versioning works and how to specify the desired API version in requests.

API Quickstart

In this section, you'll find a step-by-step guide to quickly start using the API.

Prerequisites

List any prerequisites or dependencies that users need to have in place before they can start using the API.

- Pre-requisite one
- Pre-requisite two

Authentication

Explain how to authenticate with the API, including obtaining API keys or tokens.

Making Your First Request

Provide a simple example of making a request to one of the API's endpoints. Use clear and concise code snippets.

```
GET /api/endpoint HTTP/1.1  
Host: api.example.com  
Authorization: Bearer YOUR_ACCESS_TOKEN
```

Response Handling

Explain how to handle the API responses, including parsing JSON data and handling errors.

API Usage Tips


Offer tips and best practices for using the API effectively and efficiently.

Next Steps

Suggest what users can do next, such as exploring more endpoints or integrating the API into their applications.

API Reference

This is a sample Pet Store Server based on the OpenAPI 3.0 specification.

 A very important note about this API.

Create user

POST /user

This can only be done by the logged in user.

Request parameters

Body

application/json User

Child attributes

id integer int64

username string

firstName string

lastName string

email string

password string

phone string

userStatus integer int32

User Status

JSON example

```
{
  "id": 10,
  "username": "theUser",
  "firstName": "John",
  "lastName": "James",
  "email": "john@email.com",
  "password": "12345",
  "phone": "12345",
  "userStatus": 1
}
```

Responses

default successful operation

Content type application/json

successful operation

application/json User

Child attributes

id integer int64

username string

firstName string

lastName string

email string

password string

phone string

userStatus integer int32

User Status

JSON example

```
{  
  "id": 11,  
  "username": "theUser",  
  "firstName": "John",  
  "lastName": "Doe",  
  "email": "john@email.com",  
  "password": "12345",  
  "phone": "12345",  
  "userStatus": 1  
}
```


Create list of users with given input array

POST /user/createWithList

Creates list of users with given input array

Request parameters

Body

application/json Array of User

Child attributes

id integer int64

username string

firstName string

lastName string

email string

password string

phone string

userStatus integer int32

User Status

JSON example

```
[
  {
    "id": 10,
    "username": "theUser",
    "firstName": "John",
    "lastName": "James",
    "email": "john@email.com",
    "password": "12345",
    "phone": "12345",
    "userStatus": 1
  }
]
```

Responses

200 Successful operation

Content type application/json

Successful operation

application/json User

Child attributes

id integer int64

username string

firstName string

lastName string

email string

password string

phone string

userStatus integer int32

User Status

JSON example

```
{  
  "id": 10,  
  "username": "theUser",  
  "firstName": "John",  
  "lastName": "James",  
  "email": "john@email.com",  
  "password": "12345",  
  "phone": "12345",  
  "userStatus": 1  
}
```

default successful operation

Content type

successful operation

400 Invalid username supplied

Content type application/json

Invalid username supplied

application/json ErrorResponse

Child attributes

code integer int32

details Array of object

Child attributes

typeUrl string

value string

message string

JSON example

```
{
  "code": 0,
  "details": [
    {
      "typeUrl": "string",
      "value": "string"
    }
  ],
  "message": "string"
}
```

Log user into the system

GET /user/login

Request parameters

Query

username string

The user name for login

password string

The password for login in clear text

Responses

200 successful operation

Content type application/json

successful operation

application/json string

X-Rate-Limit (Header) integer int32 **required**

calls per hour allowed by the user

X-Expires-After (Header) string date-time **required**

date in UTC when token expires

JSON example

"example"

400 Invalid username/password supplied

Content type

Invalid username/password supplied

Log current user out

GET /user/logout

Responses

default successful operation

Content type

successful operation

Find user

GET /user/{username}

Request parameters

Path

username string **required**

The name that needs to be fetched. Use user1 for testing.

Responses

200 successful operation

Content type application/json

successful operation

application/json User

Child attributes

id integer int64

username string

firstName string

lastName string

email string

password string

phone string

userStatus integer int32

User Status

JSON example

```
{  
  "id": 10,  
  "username": "theUser",  
  "firstName": "John",  
  "lastName": "James",  
  "email": "john@email.com",  
  "password": "12345",  
  "phone": "12345",  
  "userStatus": 1  
}
```

400 Invalid username supplied

Content type

Invalid username supplied

404 User not found

Content type

User not found

Update user

PUT /user/{username}

This can only be done by the logged in user.

Request parameters

Path

username string **required**

name that needs to be updated

Body

application/json User

Child attributes

id integer int64

username string

firstName string

lastName string

email string

password string

phone string

userStatus integer int32

User Status

JSON

```
{
  "id": 10,
  "username": "theUser",
  "firstName": "John",
  "lastName": "James",
  "email": "john@email.com",
  "password": "12345",
  "phone": "12345",
  "userStatus": 1
}
```

JavaScript

```
const userPayload = {
  id: 10,
  username: "theUser",
  firstName: "John",
  lastName: "James",
  email: "john@email.com",
  password: "12345",
  phone: "12345",
  userStatus: 1
};
console.log(userPayload);
```

Responses

default successful operation

Content type

successful operation

Delete user

DELETE /user/{username}

This can only be done by the logged in user.

Request parameters

Path

username string **required**

The name that needs to be deleted

Responses

400 Invalid username supplied

Content type

Invalid username supplied

404 User not found

Content type

User not found