

Swagger Petstore YAML

Table of Contents

- 1. Overview 1
 - 1.1. Version information 1
 - 1.2. Contact information..... 1
 - 1.3. License information..... 1
 - 1.4. URI scheme..... 1
 - 1.5. Tags 1
- 2. Security..... 2
 - 2.1. petstore_auth..... 2
 - 2.2. api_key 2
- 3. Paths 3
 - 3.1. Add a new pet to the store 3
 - 3.1.1. Parameters 3
 - 3.1.2. Responses 3
 - 3.1.3. Consumes 3
 - 3.1.4. Produces 3
 - 3.1.5. Tags 3
 - 3.1.6. Security 3
 - 3.2. Update an existing pet..... 3
 - 3.2.1. Parameters 4
 - 3.2.2. Responses 4
 - 3.2.3. Consumes 4
 - 3.2.4. Produces 4
 - 3.2.5. Tags 4
 - 3.2.6. Security 4
 - 3.3. Finds Pets by status 4
 - 3.3.1. Description 5
 - 3.3.2. Parameters 5
 - 3.3.3. Responses 5
 - 3.3.4. Consumes 5
 - 3.3.5. Produces 5
 - 3.3.6. Tags 5
 - 3.3.7. Security 5
 - 3.4. Finds Pets by tags 5
 - 3.4.1. Description 6
 - 3.4.2. Parameters 6
 - 3.4.3. Responses 6

3.4.4. Produces	6
3.4.5. Tags	6
3.4.6. Security	6
3.5. Updates a pet in the store with form data	6
3.5.1. Parameters	6
3.5.2. Responses	7
3.5.3. Consumes	7
3.5.4. Produces	7
3.5.5. Tags	7
3.5.6. Security	7
3.6. Find pet by ID	7
3.6.1. Description	7
3.6.2. Parameters	7
3.6.3. Responses	8
3.6.4. Consumes	8
3.6.5. Produces	8
3.6.6. Tags	8
3.6.7. Security	8
3.7. Deletes a pet	8
3.7.1. Parameters	8
3.7.2. Responses	9
3.7.3. Consumes	9
3.7.4. Produces	9
3.7.5. Tags	9
3.7.6. Security	9
3.8. uploads an image	9
3.8.1. Parameters	9
3.8.2. Responses	10
3.8.3. Consumes	10
3.8.4. Produces	10
3.8.5. Tags	10
3.8.6. Security	10
3.9. Returns pet inventories by status	10
3.9.1. Description	10
3.9.2. Responses	10
3.9.3. Produces	11
3.9.4. Tags	11
3.9.5. Security	11

3.10. Place an order for a pet	11
3.10.1. Parameters	11
3.10.2. Responses	11
3.10.3. Produces	11
3.10.4. Tags	12
3.11. Find purchase order by ID	12
3.11.1. Description	12
3.11.2. Parameters	12
3.11.3. Responses	12
3.11.4. Produces	12
3.11.5. Tags	12
3.12. Delete purchase order by ID	12
3.12.1. Description	13
3.12.2. Parameters	13
3.12.3. Responses	13
3.12.4. Produces	13
3.12.5. Tags	13
3.13. Create user	13
3.13.1. Description	13
3.13.2. Parameters	13
3.13.3. Responses	14
3.13.4. Produces	14
3.13.5. Tags	14
3.14. Creates list of users with given input array	14
3.14.1. Parameters	14
3.14.2. Responses	14
3.14.3. Produces	14
3.14.4. Tags	14
3.15. Creates list of users with given input array	15
3.15.1. Parameters	15
3.15.2. Responses	15
3.15.3. Produces	15
3.15.4. Tags	15
3.16. Logs user into the system	15
3.16.1. Parameters	15
3.16.2. Responses	16
3.16.3. Produces	16
3.16.4. Tags	16

3.17. Logs out current logged in user session	16
3.17.1. Responses	16
3.17.2. Produces	16
3.17.3. Tags	16
3.18. Get user by user name	16
3.18.1. Parameters	17
3.18.2. Responses	17
3.18.3. Produces	17
3.18.4. Tags	17
3.19. Updated user	17
3.19.1. Description	17
3.19.2. Parameters	17
3.19.3. Responses	18
3.19.4. Produces	18
3.19.5. Tags	18
3.20. Delete user	18
3.20.1. Description	18
3.20.2. Parameters	18
3.20.3. Responses	18
3.20.4. Produces	19
3.20.5. Tags	19
4. Definitions	20
4.1. ApiResponse	20
4.2. Category	20
4.3. Order	20
4.4. Pet	21
4.5. Tag	21
4.6. User	21

Chapter 1. Overview

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io>; or on irc.freenode.net, #swagger. For this sample, you can use the api key "special-key" to test the authorization filters

1.1. Version information

Version : 1.0.0

1.2. Contact information

Contact Email : apiteam@swagger.io

1.3. License information

License : Apache 2.0

License URL : <http://www.apache.org/licenses/LICENSE-2.0.html>

Terms of service : <http://swagger.io/terms/>

1.4. URI scheme

BasePath : /v2

Schemes : HTTP

1.5. Tags

- pet : Everything about your Pets
- store : Operations about user
- user : Access to Petstore orders

Chapter 2. Security

2.1. petstore_auth

Type : oauth2

Flow : implicit

Token URL : <http://petstore.swagger.io/api/oauth/dialog>

Name	Description
write:pets	modify pets in your account
read:pets	read your pets

2.2. api_key

Type : apiKey

Name : api_key

In : HEADER

Chapter 3. Paths

3.1. Add a new pet to the store

POST /v2/pet

3.1.1. Parameters

Type	Name	Description	Schema
Body	body <i>optional</i>	Pet object that needs to be added to the store	Pet

3.1.2. Responses

HTTP Code	Description	Schema
405	Invalid input	No Content

3.1.3. Consumes

- `application/json`
- `application/xml`

3.1.4. Produces

- `application/xml`
- `application/json`

3.1.5. Tags

- `pet`

3.1.6. Security

Type	Name	Scopes
oauth2	petstore_auth	write:pets,read:pets

3.2. Update an existing pet

PUT /v2/pet

3.2.1. Parameters

Type	Name	Description	Schema
Body	body <i>optional</i>	Pet object that needs to be added to the store	Pet

3.2.2. Responses

HTTP Code	Description	Schema
400	Invalid ID supplied	No Content
404	Pet not found	No Content
405	Validation exception	No Content

3.2.3. Consumes

- `application/json`
- `application/xml`

3.2.4. Produces

- `application/xml`
- `application/json`

3.2.5. Tags

- pet

3.2.6. Security

Type	Name	Scopes
oauth2	petstore_auth	write:pets,read:pets

3.3. Finds Pets by status

GET /v2/pet/findByStatus

3.3.1. Description

Multiple status values can be provided with comma separated strings

3.3.2. Parameters

Type	Name	Description	Schema	Default
Query	status <i>optional</i>	Status values that need to be considered for filter	< string > array(multi)	"available"

3.3.3. Responses

HTTP Code	Description	Schema
200	successful operation	< Pet > array
400	Invalid status value	No Content

3.3.4. Consumes

- application/xml
- application/json
- multipart/form-data
- application/x-www-form-urlencoded

3.3.5. Produces

- application/xml
- application/json

3.3.6. Tags

- pet

3.3.7. Security

Type	Name	Scopes
oauth2	petstore_auth	write:pets,read:pets

3.4. Finds Pets by tags

```
GET /v2/pet/findByTags
```

3.4.1. Description

Multiple tags can be provided with comma separated strings. Use tag1, tag2, tag3 for testing.

3.4.2. Parameters

Type	Name	Description	Schema
Query	tags <i>optional</i>	Tags to filter by	< string > array(multi)

3.4.3. Responses

HTTP Code	Description	Schema
200	successful operation	< Pet > array
400	Invalid tag value	No Content

3.4.4. Produces

- [application/xml](#)
- [application/json](#)

3.4.5. Tags

- pet

3.4.6. Security

Type	Name	Scopes
oauth2	petstore_auth	write:pets,read:pets

3.5. Updates a pet in the store with form data

```
POST /v2/pet/{petId}
```

3.5.1. Parameters

Type	Name	Description	Schema
Path	petId <i>required</i>	ID of pet that needs to be updated	string

Type	Name	Description	Schema
FormData	name <i>optional</i>	Updated name of the pet	string
FormData	status <i>optional</i>	Updated status of the pet	string

3.5.2. Responses

HTTP Code	Description	Schema
405	Invalid input	No Content

3.5.3. Consumes

- `application/x-www-form-urlencoded`

3.5.4. Produces

- `application/xml`
- `application/json`

3.5.5. Tags

- pet

3.5.6. Security

Type	Name	Scopes
oauth2	petstore_auth	write:pets,read:pets

3.6. Find pet by ID

```
GET /v2/pet/{petId}
```

3.6.1. Description

Returns a single pet

3.6.2. Parameters

Type	Name	Description	Schema
Path	petId <i>required</i>	ID of pet to return	integer (int64)

3.6.3. Responses

HTTP Code	Description	Schema
200	successful operation	Pet
400	Invalid ID supplied	No Content
404	Pet not found	No Content

3.6.4. Consumes

- `application/x-www-form-urlencoded`

3.6.5. Produces

- `application/xml`
- `application/json`

3.6.6. Tags

- pet

3.6.7. Security

Type	Name	Scopes
apiKey	api_key	
oauth2	petstore_auth	write:pets,read:pets

3.7. Deletes a pet

```
DELETE /v2/pet/{petId}
```

3.7.1. Parameters

Type	Name	Description	Schema
Header	api_key <i>optional</i>		string
Path	petId <i>required</i>	Pet id to delete	integer (int64)

3.7.2. Responses

HTTP Code	Description	Schema
400	Invalid pet value	No Content

3.7.3. Consumes

- `multipart/form-data`
- `application/x-www-form-urlencoded`

3.7.4. Produces

- `application/xml`
- `application/json`

3.7.5. Tags

- pet

3.7.6. Security

Type	Name	Scopes
oauth2	petstore_auth	write:pets,read:pets

3.8. uploads an image

```
POST /v2/pet/{petId}/uploadImage
```

3.8.1. Parameters

Type	Name	Description	Schema
Path	petId <i>required</i>	ID of pet to update	integer (int64)

Type	Name	Description	Schema
FormData	additionalMetadata <i>optional</i>	Additional data to pass to server	string
FormData	file <i>optional</i>	file to upload	file

3.8.2. Responses

HTTP Code	Description	Schema
200	successful operation	ApiResponse

3.8.3. Consumes

- `multipart/form-data`

3.8.4. Produces

- `application/json`

3.8.5. Tags

- pet

3.8.6. Security

Type	Name	Scopes
oauth2	petstore_auth	write:pets,read:pets

3.9. Returns pet inventories by status

```
GET /v2/store/inventory
```

3.9.1. Description

Returns a map of status codes to quantities

3.9.2. Responses

HTTP Code	Description	Schema
200	successful operation	< string, integer (int32) > map

3.9.3. Produces

- `application/json`

3.9.4. Tags

- store

3.9.5. Security

Type	Name
apiKey	api_key

3.10. Place an order for a pet

POST /v2/store/order

3.10.1. Parameters

Type	Name	Description	Schema
Body	body <i>optional</i>	order placed for purchasing the pet	Order

3.10.2. Responses

HTTP Code	Description	Schema
200	successful operation	Order
400	Invalid Order	No Content

3.10.3. Produces

- `application/xml`
- `application/json`

3.10.4. Tags

- store

3.11. Find purchase order by ID

```
GET /v2/store/order/{orderId}
```

3.11.1. Description

For valid response try integer IDs with value ≤ 5 or > 10 . Other values will generated exceptions

3.11.2. Parameters

Type	Name	Description	Schema
Path	orderId <i>required</i>	ID of pet that needs to be fetched	string

3.11.3. Responses

HTTP Code	Description	Schema
200	successful operation	Order
400	Invalid ID supplied	No Content
404	Order not found	No Content

3.11.4. Produces

- `application/xml`
- `application/json`

3.11.5. Tags

- store

3.12. Delete purchase order by ID

```
DELETE /v2/store/order/{orderId}
```

3.12.1. Description

For valid response try integer IDs with value < 1000. Anything above 1000 or nonintegers will generate API errors

3.12.2. Parameters

Type	Name	Description	Schema
Path	orderId <i>required</i>	ID of the order that needs to be deleted	string

3.12.3. Responses

HTTP Code	Description	Schema
400	Invalid ID supplied	No Content
404	Order not found	No Content

3.12.4. Produces

- `application/xml`
- `application/json`

3.12.5. Tags

- store

3.13. Create user

POST /v2/user

3.13.1. Description

This can only be done by the logged in user.

3.13.2. Parameters

Type	Name	Description	Schema
Body	body <i>optional</i>	Created user object	User

3.13.3. Responses

HTTP Code	Description	Schema
default	successful operation	No Content

3.13.4. Produces

- `application/xml`
- `application/json`

3.13.5. Tags

- user

3.14. Creates list of users with given input array

POST /v2/user/createWithArray

3.14.1. Parameters

Type	Name	Description	Schema
Body	body <i>optional</i>	List of user object	< User > array

3.14.2. Responses

HTTP Code	Description	Schema
default	successful operation	No Content

3.14.3. Produces

- `application/xml`
- `application/json`

3.14.4. Tags

- user

3.15. Creates list of users with given input array

POST /v2/user/createWithList

3.15.1. Parameters

Type	Name	Description	Schema
Body	body <i>optional</i>	List of user object	< User > array

3.15.2. Responses

HTTP Code	Description	Schema
default	successful operation	No Content

3.15.3. Produces

- [application/xml](#)
- [application/json](#)

3.15.4. Tags

- user

3.16. Logs user into the system

GET /v2/user/login

3.16.1. Parameters

Type	Name	Description	Schema
Query	password <i>optional</i>	The password for login in clear text	string
Query	username <i>optional</i>	The user name for login	string

3.16.2. Responses

HTTP Code	Description	Schema
200	successful operation Headers : X-Rate-Limit (integer (int32)) : calls per hour allowed by the user. X-Expires-After (string (date-time)) : date in UTC when token expires.	string
400	Invalid username/password supplied	No Content

3.16.3. Produces

- `application/xml`
- `application/json`

3.16.4. Tags

- user

3.17. Logs out current logged in user session

GET /v2/user/logout

3.17.1. Responses

HTTP Code	Description	Schema
default	successful operation	No Content

3.17.2. Produces

- `application/xml`
- `application/json`

3.17.3. Tags

- user

3.18. Get user by user name

```
GET /v2/user/{username}
```

3.18.1. Parameters

Type	Name	Description	Schema
Path	username <i>required</i>	The name that needs to be fetched. Use user1 for testing.	string

3.18.2. Responses

HTTP Code	Description	Schema
200	successful operation	User
400	Invalid username supplied	No Content
404	User not found	No Content

3.18.3. Produces

- `application/xml`
- `application/json`

3.18.4. Tags

- user

3.19. Updated user

```
PUT /v2/user/{username}
```

3.19.1. Description

This can only be done by the logged in user.

3.19.2. Parameters

Type	Name	Description	Schema
Path	username <i>required</i>	name that need to be deleted	string

Type	Name	Description	Schema
Body	body <i>optional</i>	Updated user object	User

3.19.3. Responses

HTTP Code	Description	Schema
400	Invalid user supplied	No Content
404	User not found	No Content

3.19.4. Produces

- `application/xml`
- `application/json`

3.19.5. Tags

- user

3.20. Delete user

```
DELETE /v2/user/{username}
```

3.20.1. Description

This can only be done by the logged in user.

3.20.2. Parameters

Type	Name	Description	Schema
Path	username <i>required</i>	The name that needs to be deleted	string

3.20.3. Responses

HTTP Code	Description	Schema
400	Invalid username supplied	No Content
404	User not found	No Content

3.20.4. Produces

- `application/xml`
- `application/json`

3.20.5. Tags

- `user`

Chapter 4. Definitions

4.1. ApiResponse

Name	Schema
code <i>optional</i>	integer (int32)
message <i>optional</i>	string
type <i>optional</i>	string

4.2. Category

Name	Schema
id <i>optional</i>	integer (int64)
name <i>optional</i>	string

4.3. Order

Name	Description	Schema
complete <i>optional</i>		boolean
id <i>optional</i>		integer (int64)
petId <i>optional</i>		integer (int64)
quantity <i>optional</i>		integer (int32)
shipDate <i>optional</i>		string (date-time)
status <i>optional</i>	Order Status	enum (placed, approved, delivered)

4.4. Pet

Name	Description	Schema
category <i>optional</i>		Category
id <i>optional</i>		integer (int64)
name <i>required</i>	Example : "doggie"	string
photoUrls <i>required</i>		< string > array
status <i>optional</i>	pet status in the store	enum (available, pending, sold)
tags <i>optional</i>		< Tag > array

4.5. Tag

Name	Schema
id <i>optional</i>	integer (int64)
name <i>optional</i>	string

4.6. User

Name	Description	Schema
email <i>optional</i>		string
firstName <i>optional</i>		string
id <i>optional</i>		integer (int64)
lastName <i>optional</i>		string
password <i>optional</i>		string

Name	Description	Schema
phone <i>optional</i>		string
userStatus <i>optional</i>	User Status	integer (int32)
username <i>optional</i>		string