# Swagger Petstore API

Version: 1.0.7

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

**Base URL:** https://petstore.swagger.io/v2

## Authentication

This API uses the following authentication schemes: - API Key (api\_key): API key authentication - OAuth 2.0 (petstore\_auth): Implicit flow with the following scopes: - read:pets: read your pets - write:pets: modify pets in your account

## Endpoints

### Pet

#### Upload an image

* **POST** /pet/{petId}/uploadImage
* **Description:** Uploads an image
* **Parameters:**
  + petId (path, required): ID of pet to update
  + additionalMetadata (formData, optional): Additional data to pass to server
  + file (formData, optional): file to upload
* **Responses:**
  + 200: successful operation

#### Add a new pet to the store

* **POST** /pet
* **Description:** Add a new pet to the store
* **Parameters:**
  + body (body, required): Pet object that needs to be added to the store
* **Responses:**
  + 405: Invalid input

#### Update an existing pet

* **PUT** /pet
* **Description:** Update an existing pet
* **Parameters:**
  + body (body, required): Pet object that needs to be added to the store
* **Responses:**
  + 400: Invalid ID supplied
  + 404: Pet not found
  + 405: Validation exception

#### Finds Pets by status

* **GET** /pet/findByStatus
* **Description:** Multiple status values can be provided with comma separated strings
* **Parameters:**
  + status (query, required): Status values that need to be considered for filter
* **Responses:**
  + 200: successful operation
  + 400: Invalid status value

#### Finds Pets by tags (deprecated)

* **GET** /pet/findByTags
* **Description:** Multiple tags can be provided with comma separated strings. Use tag1, tag2, tag3 for testing.
* **Parameters:**
  + tags (query, required): Tags to filter by
* **Responses:**
  + 200: successful operation
  + 400: Invalid tag value

#### Find pet by ID

* **GET** /pet/{petId}
* **Description:** Returns a single pet
* **Parameters:**
  + petId (path, required): ID of pet to return
* **Responses:**
  + 200: successful operation
  + 400: Invalid ID supplied
  + 404: Pet not found

#### Updates a pet in the store with form data

* **POST** /pet/{petId}
* **Description:** Updates a pet in the store with form data
* **Parameters:**
  + petId (path, required): ID of pet that needs to be updated
  + name (formData, optional): Updated name of the pet
  + status (formData, optional): Updated status of the pet
* **Responses:**
  + 405: Invalid input

#### Deletes a pet

* **DELETE** /pet/{petId}
* **Description:** Deletes a pet
* **Parameters:**
  + api\_key (header, optional): API key
  + petId (path, required): Pet id to delete
* **Responses:**
  + 400: Invalid ID supplied
  + 404: Pet not found

### Store

#### Returns pet inventories by status

* **GET** /store/inventory
* **Description:** Returns a map of status codes to quantities
* **Responses:**
  + 200: successful operation

#### Place an order for a pet

* **POST** /store/order
* **Description:** Place an order for a pet
* **Parameters:**
  + body (body, required): order placed for purchasing the pet
* **Responses:**
  + 200: successful operation
  + 400: Invalid Order

#### Find purchase order by ID

* **GET** /store/order/{orderId}
* **Description:** For valid response try integer IDs with value >= 1 and <= 10. Other values will generated exceptions
* **Parameters:**
  + orderId (path, required): ID of pet that needs to be fetched
* **Responses:**
  + 200: successful operation
  + 400: Invalid ID supplied
  + 404: Order not found

#### Delete purchase order by ID

* **DELETE** /store/order/{orderId}
* **Description:** For valid response try integer IDs with positive integer value. Negative or non-integer values will generate API errors
* **Parameters:**
  + orderId (path, required): ID of the order that needs to be deleted
* **Responses:**
  + 400: Invalid ID supplied
  + 404: Order not found

### User

#### Create user

* **POST** /user
* **Description:** This can only be done by the logged in user.
* **Parameters:**
  + body (body, required): Created user object
* **Responses:**
  + default: successful operation

#### Creates list of users with given input array

* **POST** /user/createWithArray
* **Description:** Creates list of users with given input array
* **Parameters:**
  + body (body, required): List of user object
* **Responses:**
  + default: successful operation

#### Creates list of users with given input array

* **POST** /user/createWithList
* **Description:** Creates list of users with given input array
* **Parameters:**
  + body (body, required): List of user object
* **Responses:**
  + default: successful operation

#### Logs user into the system

* **GET** /user/login
* **Description:** Logs user into the system
* **Parameters:**
  + username (query, required): The user name for login
  + password (query, required): The password for login in clear text
* **Responses:**
  + 200: successful operation
  + 400: Invalid username/password supplied

#### Logs out current logged in user session

* **GET** /user/logout
* **Description:** Logs out current logged in user session
* **Responses:**
  + default: successful operation

#### Get user by user name

* **GET** /user/{username}
* **Description:** Get user by user name
* **Parameters:**
  + username (path, required): The name that needs to be fetched. Use user1 for testing.
* **Responses:**
  + 200: successful operation
  + 400: Invalid username supplied
  + 404: User not found

#### Updated user

* **PUT** /user/{username}
* **Description:** This can only be done by the logged in user.
* **Parameters:**
  + username (path, required): name that need to be updated
  + body (body, required): Updated user object
* **Responses:**
  + 400: Invalid user supplied
  + 404: User not found

#### Delete user

* **DELETE** /user/{username}
* **Description:** This can only be done by the logged in user.
* **Parameters:**
  + username (path, required): The name that needs to be deleted
* **Responses:**
  + 400: Invalid username supplied
  + 404: User not found

## Models

### ApiResponse

* code (integer, int32)
* type (string)
* message (string)

### Category

* id (integer, int64)
* name (string)

### Pet

* id (integer, int64)
* category (Category)
* name (string, required)
* photoUrls (array of strings, required)
* tags (array of Tag)
* status (string, enum: available, pending, sold)

### Tag

* id (integer, int64)
* name (string)

### Order

* id (integer, int64)
* petId (integer, int64)
* quantity (integer, int32)
* shipDate (string, date-time)
* status (string, enum: placed, approved, delivered)
* complete (boolean)

### User

* id (integer, int64)
* username (string)
* firstName (string)
* lastName (string)
* email (string)
* password (string)
* phone (string)
* userStatus (integer, int32)

## External Documentation

Find out more about Swagger: <http://swagger.io>