



# **PSdemo Developer's Guide**

*1.0 (alpha)*

**Pulse Secure, LLC**

**Dec 09, 2019**

Pulse Secure, LLC  
2700 Zanker Road, Suite 200  
San Jose, CA 95134  
<https://www.pulsesecure.net>

© 2019 by Pulse Secure, LLC. All rights reserved

# CONTENTS

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>API for some functionality</b>	<b>5</b>
2.1	Pet Store APIs . . . . .	5
<b>3</b>	<b>Example Code</b>	<b>13</b>
3.1	Using API for xyz use case . . . . .	13
3.2	Using API for mno use case . . . . .	13



TBD

- *Introduction*
- *API for some functionality*
  - *Pet Store APIs*
    - \* *pet*
    - \* *store*
    - \* *user*
- *Example Code*
  - *Using API for xyz use case*
  - *Using API for mno use case*



## INTRODUCTION

TBD





## API FOR SOME FUNCTIONALITY

TBD

### 2.1 Pet Store APIs

#### 2.1.1 pet

---

##### GET /pet/{petId}

Find pet by ID

- **Description:** Returns a single pet
- **Produces:** ['application/json', 'application/xml']

##### Parameters

Name	Position	Description	Type
petId	path	ID of pet to return	integer

##### Responses

200 - successful operation

400 - Invalid ID supplied

404 - Pet not found

---

---

##### POST /pet/{petId}

Updates a pet in the store with form data

- **Description:**
- **Consumes:** ['application/x-www-form-urlencoded']
- **Produces:** ['application/json', 'application/xml']

##### Parameters

Name	Position	Description	Type
petId	path	ID of pet that needs to be updated	integer
name	formData	Updated name of the pet	string
status	formData	Updated status of the pet	string

**Responses**

*405 - Invalid input*

---

**DELETE /pet/{petId}**

Deletes a pet

- **Description:**
- **Produces:** ['application/json', 'application/xml']

**Parameters**

Name	Position	Description	Type
api_key	header		string
petId	path	Pet id to delete	integer

**Responses**

*400 - Invalid ID supplied*

*404 - Pet not found*

---

**POST /pet/{petId}/uploadImage**

uploads an image

- **Description:**
- **Consumes:** ['multipart/form-data']
- **Produces:** ['application/json']

**Parameters**

Name	Position	Description	Type
petId	path	ID of pet to update	integer
additionalMetadata	formData	Additional data to pass to server	string
file	formData	file to upload	file

**Responses**

*200 - successful operation*

---

**POST /pet**

Add a new pet to the store

- **Description:**
-

- **Consumes:** ['application/json', 'application/xml']
- **Produces:** ['application/json', 'application/xml']

**Parameters**

Name	Position	Description	Type
body	body	Pet object that needs to be added to the store	

**Responses**

405 - *Invalid input*

---

**PUT /pet**

Update an existing pet

- **Description:**
- **Consumes:** ['application/json', 'application/xml']
- **Produces:** ['application/json', 'application/xml']

**Parameters**

Name	Position	Description	Type
body	body	Pet object that needs to be added to the store	

**Responses**

400 - *Invalid ID supplied*

404 - *Pet not found*

405 - *Validation exception*

---

**GET /pet/findByStatus**

Finds Pets by status

- **Description:** Multiple status values can be provided with comma separated strings
- **Produces:** ['application/json', 'application/xml']

**Parameters**

Name	Position	Description	Type
status	query	Status values that need to be considered for filter	array

**Responses**

200 - *successful operation*

400 - *Invalid status value*

---

**GET /pet/findByTags**

Finds Pets by tags

- **Description:** Multiple tags can be provided with comma separated strings. Use tag1, tag2, tag3 for testing.
- **Produces:** ['application/json', 'application/xml']

### Parameters

Name	Position	Description	Type
tags	query	Tags to filter by	array

### Responses

200 - *successful operation*

400 - *Invalid tag value*

---

## 2.1.2 store

---

### GET /store/inventory

Returns pet inventories by status

- **Description:** Returns a map of status codes to quantities
- **Produces:** ['application/json']

### Parameters

Name	Position	Description	Type
------	----------	-------------	------

### Responses

200 - *successful operation*

---

### GET /store/order/{orderId}

Find purchase order by ID

- **Description:** For valid response try integer IDs with value  $\geq 1$  and  $\leq 10$ . Other values will generated exceptions
- **Produces:** ['application/json', 'application/xml']

### Parameters

Name	Position	Description	Type
orderId	path	ID of pet that needs to be fetched	integer

### Responses

200 - *successful operation*

400 - *Invalid ID supplied*

404 - *Order not found*

---

---

**DELETE /store/order/{orderId}**

Delete purchase order by ID

- **Description:** For valid response try integer IDs with positive integer value. Negative or non-integer values will generate API errors
- **Produces:** ['application/json', 'application/xml']

**Parameters**

Name	Position	Description	Type
orderId	path	ID of the order that needs to be deleted	integer

**Responses**

400 - Invalid ID supplied

404 - Order not found

---

---

**POST /store/order**

Place an order for a pet

- **Description:**
- **Consumes:** ['application/json']
- **Produces:** ['application/json', 'application/xml']

**Parameters**

Name	Position	Description	Type
body	body	order placed for purchasing the pet	

**Responses**

200 - successful operation

400 - Invalid Order

---

### 2.1.3 user

---

**POST /user**

Create user

- **Description:** This can only be done by the logged in user.
- **Consumes:** ['application/json']
- **Produces:** ['application/json', 'application/xml']

**Parameters**

Name	Position	Description	Type
body	body	Created user object	

### Responses

*default - successful operation*

---

### POST /user/createWithArray

Creates list of users with given input array

- **Description:**
- **Consumes:** ['application/json']
- **Produces:** ['application/json', 'application/xml']

### Parameters

Name	Position	Description	Type
body	body	List of user object	

### Responses

*default - successful operation*

---

### POST /user/createWithList

Creates list of users with given input array

- **Description:**
- **Consumes:** ['application/json']
- **Produces:** ['application/json', 'application/xml']

### Parameters

Name	Position	Description	Type
body	body	List of user object	

### Responses

*default - successful operation*

---

### GET /user/{username}

Get user by user name

- **Description:**
- **Produces:** ['application/json', 'application/xml']

## Parameters

Name	Position	Description	Type
username	path	The name that needs to be fetched. Use user1 for testing.	string

## Responses

200 - successful operation

400 - Invalid username supplied

404 - User not found

---

## PUT /user/{username}

Updated user

- **Description:** This can only be done by the logged in user.
- **Consumes:** ['application/json']
- **Produces:** ['application/json', 'application/xml']

## Parameters

Name	Position	Description	Type
username	path	name that need to be updated	string
body	body	Updated user object	

## Responses

400 - Invalid user supplied

404 - User not found

---

## DELETE /user/{username}

Delete user

- **Description:** This can only be done by the logged in user.
- **Produces:** ['application/json', 'application/xml']

## Parameters

Name	Position	Description	Type
username	path	The name that needs to be deleted	string

## Responses

400 - Invalid username supplied

404 - User not found

---

## GET /user/login

Logs user into the system

- **Description:**
- **Produces:** ['application/json', 'application/xml']

**Parameters**

Name	Position	Description	Type
username	query	The user name for login	string
password	query	The password for login in clear text	string

**Responses**

*200 - successful operation*

*400 - Invalid username/password supplied*

---

---

**GET /user/logout**

Logs out current logged in user session

- **Description:**
- **Produces:** ['application/json', 'application/xml']

**Parameters**

Name	Position	Description	Type
------	----------	-------------	------

**Responses**

*default - successful operation*

---



## EXAMPLE CODE

Perform the setup using the following commands

```
bash xyz.py -o <outputfile> -d <inputdir> -t <type>
```

### 3.1 Using API for xyz use case

### 3.2 Using API for mno use case