phpypam Documentation

codeaffen

Sep 06, 2021

USER DOCUMENTATION

1 phpypam: Python API client library for phpIPAM installation **1**
- 1.1 installation . . . 1
- 1.2 quick start . . . 1
- 1.3 making api connection . . . 2
- 1.4 get available controllers . . . 2
- 1.5 get an entity . . . 2
- 1.6 create an entity . . . 3
- 1.7 update an entity . . . 3
- 1.8 delete an entity . . . 4
- 1.9 possible exceptions . . . 4

2 phpypam **5**
- 2.1 phpypam package . . . 5

3 How to contribute to phpipam-ansible-modules **9**
- 3.1 Did you found a bug . . . 9
- 3.2 Did you wrote a patch for an open bug . . . 9
- 3.3 Do you want to add a new feature . . . 9
- 3.4 Do you wnat to contribute to documentation . . . 9
- 3.5 Thank you for any contribution . . . 10

4 Indices and tables **11**

Python Module Index **13**

Index **15**

CHAPTER

ONE

PHPYPAM: PYTHON API CLIENT LIBRARY FOR PHPIPAM INSTALLATION

PyPI version Codacy Badge Documentation Status

As we started to develop phpipam-ansible-modules we used an existing python library for phpIPAM API. As we needed a good error handling and we don't expect a quick fix of existing project we started to develop our own library.

1.1 installation

This library is hosted on pypi.org, so you can simply use `pip` to install it.

```
pip install phpypam
```

Alternatively you can install it from source. You need to do the following:

```
$ git clone https://github.com/codeaffen/phpypam.git
Cloning into 'phpypam'...
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 366 (delta 0), reused 0 (delta 0), pack-reused 365
Receiving objects: 100% (366/366), 88.57 KiB | 521.00 KiB/s, done.
Resolving deltas: 100% (187/187), done.
$ cd phpypam/
$ python setup.py install
```

1.2 quick start

To start using `phpypam` you simply have to write some lines of code.

```
import phpypam

pi = phpypam.api(
  url='https://ipam.example.com',
  app_id='ansible',
  username='apiuser',
  password='apiP455wd',
  ssl_verify=True
```

(continues on next page)

(continued from previous page)

```
)
pi.get_entity(controller='sections')
```

1.3 making api connection

To connect to phpIPAM API you need some parameters to authenticate against the phpIPAM instance.

Parameter | Description | Default | :——— | :———- | :—— | url | The URL to a phpIPAM instance. It includes the protocol (http or https). | | app_id | The app_id which is used for the API operations. | username | The `username` which is used to connect to API. | None | password | The `password` to authenticate `username` against API. | None | ssl_verify | Should certificate of endpoint verified or not. Useful if you use a self signed certificate. | True |

Example connect to api and request current token:

```
connection_params = dict(
url='https://ipam.example.com',
  app_id='ansible',
  username='apiuser',
  password='apiP455wd',
  ssl_verify=True
)

pi = phpypam.api(**connection_params)

token = pi.get_token()
```

First of all you create a dictionary with the connection data. This dictionary will unpacked for creating a `phpypam.api` object.

If all went well you can use the `get_token` to get the currently valid token from API.

1.4 get available controllers

To work with the phpIPAM api it is useful if you know all available controllers. To achieve this you can either read the api documentation or you can use the `controllers` method.

```
controllers = pi.controllers()
```

The method returns a set with all supported controllers.

1.5 get an entity

To get an entity the `get_entity` method has to be used.

```
get_entity(controller, controller_path=None, params=None)
```

Example get a `section` by name:

```
entity = pi.get_entity(controller='sections', controller_path='foobar')
```

This call returns a dictionary for the entity with the name `foobar`.

1.6 create an entity

To create an entity the `create_entity` method has to be used.

```
create_entity(controller, controller_path=None, data=None, params=None)
```

Example create a `section` if it does not exists:

```
my_section = dict(
    name='foobar',
    description='new section',
    permissions='{"3":"1","2":"2"}'
)

try:
    entity = pi.get_entity(controller='sections', controller_path=my_section['name'])
except PHPyPAMEntityNotFoundException:
    print('create entity')
    entity = pi.create_entity(controller='sections', data=my_section)
```

In this example first we check if the section we work on already exists. If the PHPyPAMEntityNotFoundException is raised we create the entity.

1.7 update an entity

To update an entity you have to use the `update_entity` method.

```
update_entity(controller, controller_path=None, data=None, params=None)
```

Example update a `section` if it exists:

```
my_section['description'] = 'new description'

entity = pi.get_entity(controller='sections', controller_path=my_section['name'])
pi.update_entity(controller='sections', controller_path=entity['id'], data=my_section)
```

To change data you have to modify the value of the desired key to the value you want. You can see the data is changed in the dict from the former example. Then you get the entity to obtain its id to work on.

Note: All modifying operations need the id of an entity not the name.

In the last step you call `update_entity` and put the entity id in parameter `controller_path` with the `data` parameter you provide the fully entity description dictionary.

1.8 delete an entity

To delete an entity you have to use the `delete_entity` method.

```
delete_entity(controller, controller_path, params=None)
```

Example delete a existing section:

```
entity = pi.get_entity(controller='sections', controller_path=my_section['name'])
pi.delete_entity(controller='sections', controller_path=entity['id'])
```

In this example you request the entity you had created/updated in the above examples. After that you call `delete_entity` with the entity id from the request before.

1.9 possible exceptions

- ***PHPyPAMInvalidCredentials*** - will be raised if something goes wrong with the authentication
- ***PHPyPAMEntityNotFoundException*** - will be raised if an entity does not exists
- ***PHPyPAMInvalidSyntax*** - will be raised for requests which will be answered with status code 400 from API
- ***PHPyPAMException*** - for any errors which we catch but no specific exception exists this exception wil be raised

CHAPTER

TWO

PHPYPAM

2.1 phpypam package

Package that provides phpIPAM API interface.

2.1.1 Subpackages

phpypam.core package

Core package provide api and exception classes.

Submodules

phpypam.core.api module

Default class to handle all api interactions with phpIPAM server.

class phpypam.core.api.**Api**(*url*, *app_id*, *username=None*, *password=None*, *token=None*, *encryption=False*, *timeout=None*, *ssl_verify=True*, *user_agent=None*)

Bases: `object`

The main class.

It generates tha API object where you can run different actions again to *create*, *update* and *delete* entities. It also provides functions with informational character only.

controllers()

Report all controllers from phpIPAM API.

This method is used to report all known controllers of phpIPAM API. Unfortunately the API doesn't report all nor the correct paths for all 'controllers'.

Returns Returns a tuple of controller paths.

Return type tuple

create_entity(*controller*, *controller_path=None*, *data=None*, *params=None*)

Create an entity.

Parameters

- **controller** (*str*) – Name of the controller to use.

- **controller_path** (*str, optional*) – The path which is used to query for entities, defaults to None
- **data** (*dict*) – Dictionary, list of tuples, bytes, or file-like object to send in the body of the Request.
- **params** (*dict, optional*) – Dictionary list of tuples or bytes to send in the query string for the Request., defaults to None

Returns Returns the newly created entity.

Return type Union[dict, list]

delete_entity(*controller*, *controller_path*, *params=None*)
Delete an entity.

Parameters

- **controller** (*str*) – Name of the controller to use.
- **controller_path** (*str*) – The path wich is used to access the entity to delete.
- **params** (*dict, optional*) – Dictionary, list of tuples or bytes to send in the query string for the Request., defaults to None

Returns Returns True if entity was deleted successfully or either 'dict' or 'list' of entities to work on.

Return type Union[book, dict, list]

get_entity(*controller*, *controller_path=None*, *params=None*)
Get existing entity from phpIPAM server.

This method query for existing entity. It there a result it will be returned otherwise an PhpIPAMEntityNotFound exception is raised from underlying method.

Parameters

- **controller** (*str*) – Name of the controller to request entity from.
- **controller_path** (*str, optional*) – The path which is used to query for entities, defaults to None
- **params** (*dict, optional*) – Request parameters which have to be append to the request URI, defaults to None

Returns Result of the query. It can be either a 'list' or 'dict'.

Return type Union[dict, list]

get_token()
Return last login token.

Returns Returns the api token from the last successful login.

Return type str

update_entity(*controller*, *controller_path=None*, *data=None*, *params=None*)
Update an entity.

Parameters

- **controller** (*str*) – Name of the controller to use.
- **controller_path** (*str, optional*) – The path which is used to access the entity to update., defaults to None

- **data** (*dict, optional*) – Dictionary, list of tuples, bytes, or file-like object to send in the body of the Request., defaults to None
- **params** (*dict, optional*) – Dictionary list of tuples or bytes to send in the query string for the Request., defaults to None

Returns Returns either a 'dict' or 'list' of the changed entity

Return type Union[dict, list]

phpypam.core.exceptions module

Class to provided different Exceptions.

exception phpypam.core.exceptions.**PHPyPAMEntityNotFoundException**(**args*, ***kwargs*)
Bases: Exception

Exception PHPyPAMEntityNotFoundException, children of Exception.

This Exception is raised if an entity was not found.

exception phpypam.core.exceptions.**PHPyPAMException**(**args*, *code=None*, *message=None*)
Bases: Exception

PHPyPAMExeption, children of Exception.

This exception is raised if anythings in phpypam.api doesn't work out.

exception phpypam.core.exceptions.**PHPyPAMInvalidCredentials**(**args*, ***kwargs*)
Bases: Exception

Exception PHPyPAMInvalidCredentials, children of Exception.

This Exception is raised if there are any issues with the authentication against phpIPAM api.

exception phpypam.core.exceptions.**PHPyPAMInvalidSyntax**(**args*, ***kwargs*)
Bases: Exception

Exception PHPyPAMInvalidSyntax, children of Exception.

This Exception is raised if there are any issues with syntax of request against phpIPAM api.

CHAPTER

THREE

HOW TO CONTRIBUTE TO PHPIPAM-ANSIBLE-MODULES

3.1 Did you found a bug

- Do not open Github issue it the bug concerns {php}IPAM and not the ansible modules.
- Make sure the bug is not already opened by another user.
- If you can't find an open issue which reflects your observed problem go ahead an open a new bug.
- Provide as much information as mentioned in the bug report template.

3.2 Did you wrote a patch for an open bug

- Open new pull request containing the patch.
- Provide a clear description which describes the problem and the solution. Link the existing bug to the PR.

3.3 Do you want to add a new feature

- Make sure there isn't already a feature request.
- If you can't find an open feature request which describe your feature idea or parts of it feel free to open a new feature request.
- Suggest your feature idea within the created feature request.
- Provide as much discribtion as possible to enable others to have a good understanding of what you are doing.
- Point out that you want to start to work on the new feature

3.4 Do you wnat to contribute to documentation

- Write you documentation change.
- Open a PR with your change.
- Discuss with the team about your changes.

3.5 Thank you for any contribution

We will thank you for heed the contribution guidelines and we encourage you to contribute and join the team.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

phpypam, 5
phpypam.core, 5
phpypam.core.api, 5
phpypam.core.exceptions, 7

INDEX

A

Api (*class in phpypam.core.api*), 5

C

controllers() (*phpypam.core.api.Api method*), 5
create_entity() (*phpypam.core.api.Api method*), 5

D

delete_entity() (*phpypam.core.api.Api method*), 6

G

get_entity() (*phpypam.core.api.Api method*), 6
get_token() (*phpypam.core.api.Api method*), 6

M

module
    phpypam, 5
    phpypam.core, 5
    phpypam.core.api, 5
    phpypam.core.exceptions, 7

P

phpypam
    module, 5
phpypam.core
    module, 5
phpypam.core.api
    module, 5
phpypam.core.exceptions
    module, 7
PHPyPAMEntityNotFoundException, 7
PHPyPAMException, 7
PHPyPAMInvalidCredentials, 7
PHPyPAMInvalidSyntax, 7

U

update_entity() (*phpypam.core.api.Api method*), 6