# pycic Documentation

***Release 0.1.3***

**Milton Mazzarri**

January 07, 2014

Contents

`pycic` is an MIT licensed client, written in Python, that lets you interact with the CIC (Centro de Integración Ciudadana) API.

`pycic` currently works with the `nl` and `sal` accounts. You might notice that `nl` is the default account, but you can change quickly that behavior at the moment of the instance creation, for example: `Report(account="sal")`.

CHAPTER 1

# Feature support

- Supports multiple accounts.
- Retrieves all the available Groups, Categories and Reports.
- Limits the number of reports to retrieve, also, you can filter for category or dates.
- Creates new reports.
- Supports for proxies. You can find more details in the *Examples* section.
- Are you asking yourself if `pycic` works with Python 3?, *Yes*, the following is a list of Python platforms that are officially supported:
  - Python 2.6
  - Python 2.7
  - Python 3.2
  - Python 3.3
  - PyPy

CHAPTER 2

# API Documentation

## 2.1 Report

Report class definition.

This module contains the Report class definition

**class** pycic.report.**Report**(*base_url='http://api.cic.mx'*, *version=0*, *account='nl'*, *proxies=None*)
Bases: pycic.base.BaseMethod

Retrieve and save Reports.

**get**(***kwargs*)
Retrieve reports, by default we get all the reports.

**Parameters**

- **limit** (*int or None*) – The total reports to get
- **for_category** (*int or None*) – The ID of the category to filter.
- **until** (*datetime or None*) – Show previous reports to the date indicated

**Returns** JSON object representation of the list of Reports

**Return type** dict

**Raises** TypeError, ValueError, InvalidCategory

**raise_for_category**(*field*)
Raise InvalidCategory if one ocurred for category.

**raise_for_field**(*field*, *type_of_field*)
Raise TypeError if occurred for field.

**raise_for_lat_or_lng**(*field*, *limits*)
Raise ValueError if one ocurred for latitude or longitude fields.

**raise_for_limit**()
Raise ValueError or TypeError if one occurred for limit filter.

**raise_for_return_path**()
Raise ValueError if occurred for return_path field.

**raise_for_until**()
Raise TypeError if ocurred for until filter.

**save** (***kwargs*)
Create report.

**Parameters**

- **title** (*str*) – Report title
- **content** (*str*) – Report content (required)
- **first_name** (*str*) – Report Contact, name of sender in string format
- **last_name** (*str*) – Report Contact, name of sender in string format.
- **return_path** (*str*) – Report Contact in URI format Valid schemes supported: HTTP HTTPS MAILTO
- **lat** (*float*) – The report latitude in WGS84 decimal
- **lng** (*float*) – The report longitude in WGS84 decimal
- **video_url** (*str*) – Asset for the current report, the field accepts a single string with an URL for a valid asset.
- **category** (*str*) – Send here the category name. You can get valid names from Categories (required)

**Raises** NameError, InvalidCategory, TypeError

## 2.2 Category

Category class definition.

This module contains the Category class definition

**class** pycic.category.**Category**(*base_url='http://api.cic.mx'*, *version=0*, *account='nl'*, *proxies=None*)
Bases: pycic.base.BaseMethod

Retrieve all categories available in the account.

## 2.3 Group

Group class definition.

This module contains the Group class definition

**class** pycic.group.**Group**(*base_url='http://api.cic.mx'*, *version=0*, *account='nl'*, *proxies=None*)
Bases: pycic.base.BaseMethod

Retrieve all groups available in the account.

## 2.4 Base

Base method class definition.

This module implements the Base Method for new request to the CIC's API

**class** pycic.base.**BaseMethod**(*base_url=None*, *version=None*, *account=None*, *proxies=None*)
Bases: object

Base class for the available methods in CIC's API.

**get**(**kwargs*)
Retrieve all the available entries for the method especified.

**save**()
Save reports, groups or categories via CIC's API

## 2.5 Exceptions

Custom exceptions.

This module contains the set of pycic's exceptions.

**exception** pycic.exceptions.**InvalidCategory**
Bases: exceptions.IOError

A valid category is required to make a request.

CHAPTER 3

# Examples

## 3.1 Examples

### 3.1.1 Reports

Get all the available *reports*

```
>>> from pycic.report import Report
>>> r = Report()
>>> r.get()
{u'reports': [{u'votes': 0, u'group': u'Vialidad y Transito (SS)', u'created_at': u'2013-12-22T18:09
```

But also you can `limit` the number of results, filter for categories or dates.

```
>>> from datetime import datetime
>>> from pycic.report import Report
>>> now = datetime.now()
>>> r = Report()
>>> r.get(until=now, limit=5, for_category=407)
{u'reports': [{u'votes': 0, u'group': u'Vialidad y Transito (SS)', u'created_at': u'2013-12-21T16:54
```

If you want to create a new report, it's easy, the only required attributes are `content` and `category`, but you are free to insert `title`, `first_name`, `last_name`, `return_path`, `lat`, `lng` and `video_url`.

```
>>> from pycic.report import Report
>>> r = Report()
>>> r.save(title="API Demo", content="API Demo", category="ACCIDENTE")
{u'reports': {u'votes': 0, u'group': u'', u'created_at': u'2013-12-22T23:27:07-06:00', u'updated_at'
```

### 3.1.2 Groups

You can get all the available *groups*

```
>>> from pycic.group import Group
>>> g = Group()
>>> g.get()
{u'groups': [{u'id': 402, u'categories': [[u'FALTA ELECTRICIDAD', 423]], u'name': u'CFE Golfo Norte'
```

### 3.1.3 Categories

You can get all the available *categories*

```
>>> from pycic.category import Category
>>> c = Category()
>>> c.get()
{u'categories': [{u'group': [u'Vialidad y Transito (SS)'], u'metadata': False, u'type': u'blackbox',
```

### 3.1.4 Proxies support

If you need to use a proxy, you can configure at the moment of the instance creation:

```
>>> from pycic.category import Category
>>> proxies = {
...   "http": "http://10.10.1.10:3128",
...   "https": "http://10.10.1.10:1080",
... }
>>> c = Category(proxies=proxies)
>>> result = c.get()
```

This is also applicable to *Group* and *Report* classes.

You can also configure proxies by setting the environment variables `HTTP_PROXY` and `HTTPS_PROXY`.

```
$ export HTTP_PROXY="http://10.10.1.10:3128"
$ export HTTPS_PROXY="http://10.10.1.10:1080"
```

To use HTTP Basic Auth with your proxy, use the `http://user:password@host/` syntax:

```
proxies = {
    "http": "http://user:pass@10.10.1.10:3128/",
}
```

You can find more details about proxy support in the Requests documentation.

CHAPTER 4

# Indices and tables

- *genindex*
- *modindex*
- *search*

## p