ZBX Documentation

Release 0.1.0

Xavier Barbosa

May 20, 2014

Contents

Contents:

ZBX

This library let you to describe Zabbix configuration in pure Python. This configuration can then be dumped in xml and imported into zabbix.

- Free software: BSD license
- Documentation: http://zbx.rtfd.org.

1.1 Features

1.1.1 zbx.api

```
from zb.api import *

configure(user=YOUR_USER, password=YOUR_PASSWORD, url=YOUR_URL)
reponse = request('history.get', {
    'output': 'extend',
    'history': 0,
    'itemids': '23296',
    'sortfield': 'clock',
    'sortorder': 'DESC',
    'limit': 10
})
```

1.1.2 zbx.config

```
from zb.api import *
from zb.config.items.aggregate import AvgItem

configuration = Config()
template = configuration.templates.new('My template')
classic_item = template.items.new('my item', key='my.item', applications=['my application'])
average_item = template.items.add(AvgItem('my item',
                                          groups=['first group', 'second group'],
                                          key='my.item',
                                          applications=['my application']))
```

CHAPTER 2

Installation

At the command line:

```
$ easy_install zbx
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv zbx
$ pip install zbx
```

CHAPTER 3

Simple Usage

To use ZBX in a project:

```
import zbx
```

Accessing the API

4.1 zbx.api

Access to zabbix api as described here: https://www.zabbix.com/documentation/2.2/manual/api

class `zbx.api.`**`Api`**(*user*, *password*, *url*, *auth_token=None*)
    Main api object

`authenticate`(*reset=False*)
    Authenticates to the api.

`request`(*method*, *params=None*, *auth_token=None*)
    Handle a request to the api.

It will authenticate automatically if auth_token was not provided

`zbx.api.`**`cast`**(*data*)
    Ensure that int are int etc...

`zbx.api.`**`authenticate`** **= <bound method Api.authenticate of <zbx.api.Api object at 0x28a7410>>**
    authenticate with the global api instance

`zbx.api.`**`request`** **= <bound method Api.request of <zbx.api.Api object at 0x28a7410>>**
    request with the global api instance

`zbx.api.`**`configure`**(***attrs*)
    Configure the global api instance.

CHAPTER 5

Generate the configuration

5.1 zbx.config

class `zbx.config.`**`Application`**(*name*, ***fields*)
    Application model

class `zbx.config.`**`DiscoveryRule`**(*name*, ***fields*)
    DiscoveryRule model

class `zbx.config.`**`Config`**
    Main config model

class `zbx.config.`**`Graph`**(*name*, ***fields*)
    Graph model

class `zbx.config.`**`GraphItem`**(*item=None*, ***fields*)
    GraphItem model

class `zbx.config.`**`Group`**(*name*, ***fields*)
    Group model

class `zbx.config.`**`Host`**(*name*, ***fields*)
    Host model

class `zbx.config.`**`Interface`**(*ident*, ***fields*)
    Interface model

class `zbx.config.`**`Item`**(*name*, ***fields*)
    Item model

class `zbx.config.`**`Macro`**(***fields*)
    Macro model

class `zbx.config.`**`Screen`**(*name*, ***fields*)
    Screen model

class `zbx.config.`**`ScreenItem`**(*graph=None*, ***fields*)
    ScreenItem model

class `zbx.config.`**`Template`**(*name*, ***fields*)
    Template model

class `zbx.config.`**`Trigger`**(*name*, ***fields*)
    Trigger model

class `zbx.config.`**`ValueMap`**

ValueMap model

5.2 zbx.config.item.aggregate

see https://www.zabbix.com/documentation/2.0/manual/config/items/itemtypes/aggregate # NOQA

class `zbx.config.items.aggregate.`**`AggregateItem`**(*name*, *groups*, *groupfunc*, *itemfunc*, *timeperiod*, ***fields*)

AggregateItem model

`zbx.config.items.aggregate.`**`AvgItem`**(*name*, *groups*, ***fields*)

Helper for average items.

`zbx.config.items.aggregate.`**`SumItem`**(*name*, *groups*, ***fields*)

Helper for sum items.

CHAPTER 6

Dump and load configuration

6.1 zbx.io

Dump and load config from xml files

CHAPTER 7

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

7.1 Types of Contributions

7.1.1 Report Bugs

Report bugs at https://github.com/johnnoone/zbx/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

7.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

7.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

7.1.4 Write Documentation

ZBX could always use more documentation, whether as part of the official ZBX docs, in docstrings, or even on the web in blog posts, articles, and such.

7.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/johnnoone/zbx/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

7.2 Get Started!

Ready to contribute? Here's how to set up *zbx* for local development.

1. Fork the *zbx* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/zbx.git
   ```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

   ```
   $ mkvirtualenv zbx
   $ cd zbx/
   $ python setup.py develop
   ```

4. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

   ```
   $ flake8 zbx tests
   $ python setup.py test
   $ tox
   ```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

   ```
   $ git add .
   $ git commit -m "Your detailed description of your changes."
   $ git push origin name-of-your-bugfix-or-feature
   ```

7. Submit a pull request through the GitHub website.

7.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/johnnoone/zbx/pull_requests and make sure that the tests pass for all supported Python versions.

7.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_zbx
```

CHAPTER 8

Credits

8.1 Development Lead

- Xavier Barbosa <clint.northwood@gmail.com>

8.2 Contributors

None yet. Why not be the first?

CHAPTER 9

History

9.1 0.1.0 (2014-05-02)

- Starting this project.

CHAPTER 10

Indices and tables

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