python-rtkit Documentation

Release

Andrea De Marco <24erre@gmail.com>

February 02, 2013

CONTENTS

CHAPTER

ONE

REQUEST TRACKER REST INTERFACE

Best Practical RT (Request Tracker) data access python module for REST interface.

1.1 Installation

Using pip:

```
$ pip install python-rtkit
```

Using pip dev:

```
$ pip install git+https://github.com/z4r/python-rtkit
```

1.2 RT REST API Summary

More detailed version: Request Tracker Wiki

01	W	Create ticket	ticket/new
02	RW	Read/Update ticket	ticket/<ticket-id>
03	W	Create ticket comment	ticket/<ticket-id>/comment
04	RW	Read/Update ticket links	ticket/<ticket-id>/links
05	R	Read ticket attachments	ticket/<ticket-id>/attachments
06	R	Read ticket attachment	ticket/<ticket-id>/attachments/<attachment-id>
07	R	Read ticket attachment content	ticket/<ticket-id>/attachments/<attachment-id>/content
08	R	Read ticket history	ticket/<ticket-id>/history
09	R	Read detailed ticket history	ticket/<ticket-id>/history?format=l
10	R	Read ticket history item	ticket/<ticket-id>/history/id/<history-id>
11	R	Read user by id	user/<user-id>
12	R	Read user by name	user/<user-Name>
13	R	Read queue by id	queue/<queue-id>
14	R	Read queue by name	queue/<queue-Name>
15	R	Search tickets	search/ticket?query=<q>&orderby=<o>&format=<f>

1.3 Authentication Handlers

1.3.1 Basic Authentication

```
from rtkit.resource import RTResource
from rtkit.authenticators import BasicAuthenticator
from rtkit.errors import RTResourceError

from rtkit import set_logging
import logging
set_logging('debug')
logger = logging.getLogger('rtkit')

resource = RTResource('http://<HOST>/REST/1.0/', '<USER>', '<PWD>', BasicAuthenticator)
```

1.3.2 Cookie-based Authentication

```
from rtkit.resource import RTResource
from rtkit.authenticators import CookieAuthenticator
from rtkit.errors import RTResourceError

from rtkit import set_logging
import logging
set_logging('debug')
logger = logging.getLogger('rtkit')

resource = RTResource('http://<HOST>/REST/1.0/', '<USER>', '<PWD>', CookieAuthenticator)
```

1.3.3 Kerberos Authentication

```
from rtkit.resource import RTResource
from rtkit.authenticators import KerberosAuthenticator
from rtkit.errors import RTResourceError

from rtkit import set_logging
```

```
import logging
set_logging('debug')
logger = logging.getLogger('rtkit')

resource = RTResource(url, None, None, KerberosAuthenticator)
```

Warning: Remeber to install *urllib2_kerberos*.

1.4 Overview on Low Level API

1.4.1 Create ticket

```
content = {
    'content': {
        'Queue': 1,#'', 2
        'Subject' : 'New Ticket',
        'Text' : 'My useless\ntext on\nthree lines.',
    }
}
try:
    response = resource.post(path='ticket/new', payload=content,)
    logger.info(response.parsed)
except RTResourceError as e:
    logger.error(e.response.status_int)
    logger.error(e.response.status)
    logger.error(e.response.parsed)

#OK
[DEBUG] POST ticket/new
[DEBUG] {'Content-Type': 'application/x-www-form-urlencoded', 'Accept': 'text/plain', 'User-Agent': '
[DEBUG] u'content=Queue: 1\nText: My useless\n text on\n three lines.\nSubject: New Ticket\n'
[INFO] HTTP_STATUS: 200 OK
[DEBUG] 'RT/3.8.10 200 Ok\n\n# Ticket 17 created.\n\n'
[INFO] RESOURCE_STATUS: 200 Ok
[INFO] [[('id', 'ticket/17')]]

#MISSING OR MISSPELLED QUEUE
[DEBUG] POST ticket/new
[DEBUG] {'Content-Type': 'application/x-www-form-urlencoded', 'Accept': 'text/plain', 'User-Agent': '
[DEBUG] u'content=Queue: \nText: My useless\n text on\n three lines.\nSubject: New Ticket\n'
[INFO] HTTP_STATUS: 200 OK
[DEBUG] 'RT/3.8.10 200 Ok\n\n# Could not create ticket.\n# Could not create ticket. Queue not set\n\n
[INFO] RESOURCE_STATUS: 400 Could not create ticket. Queue not set
[ERROR] 400
[ERROR] 400 Could not create ticket. Queue not set
[ERROR] []

#NO PERMISSION ON QUEUE
[DEBUG] POST ticket/new
[DEBUG] {'Content-Type': 'application/x-www-form-urlencoded', 'Accept': 'text/plain', 'User-Agent': '
[DEBUG] u'content=Queue: 2\nText: My useless\n text on\n three lines.\nSubject: New Ticket\n'
[INFO] HTTP_STATUS: 200 OK
[DEBUG] "RT/3.8.10 200 Ok\n\n# Could not create ticket.\n# No permission to create tickets in the que
[INFO] RESOURCE_STATUS: 400 No permission to create tickets in the queue '___Approvals'
```

```
[ERROR] 400
[ERROR] 400 No permission to create tickets in the queue '___Approvals'
[ERROR] []
```

1.4.2 Read a ticket

```
try:
    response = resource.get(path='ticket/1')
    for r in response.parsed:
        for t in r:
            logger.info(t)
except RTResourceError as e:
    logger.error(e.response.status_int)
    logger.error(e.response.status)
    logger.error(e.response.parsed)

#TICKET FOUND
[DEBUG] GET ticket/1
[DEBUG] {'Accept': 'text/plain', 'User-Agent': 'pyRTkit/0.0.1'}
[DEBUG] None
[INFO] HTTP_STATUS: 200 OK
[DEBUG] 'RT/3.8.10 200 Ok\n\nid: ticket/1\nQueue: General\nOwner: Nobody\nCreator: pyrtkit\nSubject:
[INFO] RESOURCE_STATUS: 200 Ok
[INFO] ('id', 'ticket/1')
[INFO] ('Queue', 'General')
[INFO] ('Owner', 'Nobody')
[INFO] ('Creator', 'pyrtkit')
[INFO] ('Subject', 'pyrt-create4')
[INFO] ('Status', 'open')
[INFO] ('Priority', '5')
[INFO] ('InitialPriority', '0')
[INFO] ('FinalPriority', '0')
[INFO] ('Requestors', '')
[INFO] ('Cc', '')
[INFO] ('AdminCc', '')
[INFO] ('Created', 'Sun Jul 03 10:48:57 2011')
[INFO] ('Starts', 'Not set')
[INFO] ('Started', 'Not set')
[INFO] ('Due', 'Not set')
[INFO] ('Resolved', 'Not set')
[INFO] ('Told', 'Wed Jul 06 12:58:00 2011')
[INFO] ('LastUpdated', 'Thu Jul 07 14:42:32 2011')
[INFO] ('TimeEstimated', '0')
[INFO] ('TimeWorked', '25 minutes')
[INFO] ('TimeLeft', '0')

#TICKET NOT FOUND
[DEBUG] GET ticket/100
[DEBUG] {'Accept': 'text/plain', 'User-Agent': 'pyRTkit/0.0.1'}
[DEBUG] None
[INFO] HTTP_STATUS: 200 OK
[DEBUG] 'RT/3.8.10 200 Ok\n\n# Ticket 100 does not exist.\n\n\n'
[INFO] RESOURCE_STATUS: 404 Ticket 100 does not exist
[ERROR] 404
[ERROR] 404 Ticket 100 does not exist
[ERROR] []
```

1.4.3 Edit a ticket or ticket's links

Ticket (or ticket's links) editing hasn't all-or-nothing behaviour; so it's very difficult to capture errors. For example trying to change Queue to a not admitted one (or to edit an unknown field) RT will return:

```
RT/3.8.10 409 Syntax Error

# queue: You may not create requests in that queue.
# spam: Unknown field.

id:
Subject: Try Edit Ticket
TimeWorked: 1
Queue: 2
Spam: 10
```

For now rtkit will raise SyntaxError with the errors list in e.response.parsed

```
[DEBUG] POST ticket/1
[DEBUG] {'Content-Type': 'application/x-www-form-urlencoded', 'Accept': 'text/plain', 'User-Agent': '
[DEBUG] u'content=Queue: 2\nSpam: 10\nTimeWorked: 1\nSubject: Try Edit Ticket\n'
[INFO] HTTP_STATUS: 200 OK
[DEBUG] 'RT/3.8.10 409 Syntax Error\n\n# queue: You may not create requests in that queue.\n# spam: U
[INFO] RESOURCE_STATUS: 409 Syntax Error
[ERROR] 409
[ERROR] 409 Syntax Error
[ERROR] [[('queue', 'You may not create requests in that queue.'), ('spam', 'Unknown field.')]]
```

1.4.4 Comment on a Ticket with Attachments

Usually your requests will be something like this.

```
try:
    params = {
        'content' :{
            'Action' : 'comment',
            'Text' : 'Comment with attach',
            'Attachment' : 'x.txt, 140x105.jpg',
        },
        'attachment_1' : file('x.txt'),
        'attachment_2' : file('140x105.jpg'),
    }
    response = resource.post(path='ticket/16/comment', payload=params,)
    for r in response.parsed:
        for t in r:
            logger.info(t)
except RTResourceError as e:
    logger.error(e.response.status_int)
    logger.error(e.response.status)
    logger.error(e.response.parsed)
```

1.5 License

This software is licensed under the `Apache License 2.0`. See the `LICENSE` file in the top distribution directory for the full license text.

1.6 References

- Best Practical RT
- Request Tracker Wiki

CHAPTER

TWO

RTKIT

2.1 rtkit.authenticators

Connect to an RT server using various authentication techniques

- The `AbstractAuthenticator` contains the base methods
- **And the current implementations are:**
  - `BasicAuthenticator`
  - `CookieAuthenticator`
  - `KerberosAuthenticator`

See Also:

`rtkit.resource` for usage

class `rtkit.authenticators.`**`AbstractAuthenticator`**(*username*, *password*, *url*, **handlers*)
Bases: `object`

Abstract Base Authenticator

Parameters

- **username** – The RT Login
- **password** – Plain Text Password
- **url** – the url ?
- ***handlers** – todo

`login`()
Login to server, unless already logged in

`open`(*request*)
Open connection to server

class `rtkit.authenticators.`**`BasicAuthenticator`**(*username*, *password*, *url*)
Bases: `rtkit.authenticators.AbstractAuthenticator`

Basic Authenticator

```
from rtkit.resource import RTResource
from rtkit.authenticators import BasicAuthenticator
from rtkit.errors import RTResourceError

from rtkit import set_logging
```

```
import logging
set_logging('debug')
logger = logging.getLogger('rtkit')

resource = RTResource('http://<HOST>/REST/1.0/', '<USER>', '<PWD>', BasicAuthenticator)
```

login()
Login to server, unless already logged in

open(*request*)
Open connection to server

class rtkit.authenticators.**CookieAuthenticator**(*username*, *password*, *url*)
Bases: rtkit.authenticators.AbstractAuthenticator

Authenticate against server using a cookie

```
from rtkit.resource import RTResource
from rtkit.authenticators import CookieAuthenticator
from rtkit.errors import RTResourceError

from rtkit import set_logging
import logging
set_logging('debug')
logger = logging.getLogger('rtkit')

resource = RTResource('http://<HOST>/REST/1.0/', '<USER>', '<PWD>', CookieAuthenticator)
```

login()
Login to server, unless already logged in

open(*request*)
Open connection to server

class rtkit.authenticators.**KerberosAuthenticator**(*username*, *password*, *url*)
Bases: rtkit.authenticators.AbstractAuthenticator

Authenticate using Kerberos

Warning:
- Requires the urllib2_kerberos
- http://pypi.python.org/pypi/urllib2_kerberos/
- sudo easy_install urllib2_kerberos

```
from rtkit.resource import RTResource
from rtkit.authenticators import KerberosAuthenticator
from rtkit.errors import RTResourceError

from rtkit import set_logging
import logging
set_logging('debug')
logger = logging.getLogger('rtkit')

resource = RTResource(url, None, None, KerberosAuthenticator)
```

login()
Login to server, unless already logged in

open(*request*)
Open connection to server

2.2 rtkit.comment

exception rtkit.comment.**RTCreated**(*msg*)
Bases: exceptions.Exception

Created Exception

args

message

exception rtkit.comment.**RTNoMatch**
Bases: exceptions.Exception

No Match Exception

args

message

rtkit.comment.**check**(*section*)
Parse and Dispatch Errors

See Also:

The rtkit.errors module

```
>>> check(['# Unknown object type: spam'])
Traceback (most recent call last):
        ...
RTUnknownTypeError: Unknown object type: spam
>>> check(["# Invalid object specification: 'spam'"])
Traceback (most recent call last):
        ...
RTInvalidError: Invalid object specification: 'spam'
>>> check(['# spam 1 does not exist.'])
Traceback (most recent call last):
        ...
RTNotFoundError: spam 1 does not exist
>>> check(['# No spam named ham exists.'])
Traceback (most recent call last):
        ...
RTNotFoundError: No spam named ham exists
>>> check(['# Objects of type eggs must be specified by numeric id.'])
Traceback (most recent call last):
        ...
RTValueError: Objects of type eggs must be specified by numeric id
>>> check(['No matching results.'])
Traceback (most recent call last):
        ...
RTNoMatch: No matching results
>>> check(['# Could not create ticket.', '# Could not create ticket. Queue not set'])
Traceback (most recent call last):
        ...
RTInvalidError: Could not create ticket. Queue not set
>>> try:
...     check(['# Ticket 1 created.'])
... except RTCreated as e:
...     e.id
'ticket/1'
>>> check(['# You are not allowed to modify ticket 2.'])
```

```
Traceback (most recent call last):
        ...
RTUnauthorized: You are not allowed to modify ticket 2
```

2.3 rtkit.entities

class rtkit.entities.**RTEntity**(*id*)
Bases: object

Base Class for an Entity

static api()

Returns NotImplementedError - needs to be implemented in subclass

id

Returns int with the ID

class rtkit.entities.**User**(*id*, ***kwargs*)
Bases: rtkit.entities.RTEntity

User Object

static api()

Returns str with 'user'

id

Returns int with the ID

class rtkit.entities.**Queue**(*id*, ***kwargs*)
Bases: rtkit.entities.RTEntity

Queue Object

static api()

Returns str with 'queue'

description = None
Queue Description

id

Returns int with the ID

name = None
Queue Name

class rtkit.entities.**Ticket**(*id*, ***kwargs*)
Bases: rtkit.entities.RTEntity

Ticket Object

static api()

Returns str with 'ticket'

creator = None
Creator

date = None
Dates as dicionary with keys

- •created
- •started
- •due
- •resolved
- •updated

delta = None
Time Deltas dictionary with keys

- •worked
- •estimated
- •left

id

Returns int with the ID

owner = None
Owner

priority = None
Priority

status = None
Status

subject = None
Subject

class rtkit.entities.**Attachment**(*id*, ***kwargs*)
Bases: rtkit.entities.RTEntity

Attachment Object

static api()

Returns str with 'attachments'

content = None
Content

ctype = None
ContentType

encoding = None
ContentEncoding

filename = None
Filename

id

class rtkit.entities.**History**(*id*)
Bases: rtkit.entities.RTEntity

History Object

Todo

History not implemented

---

static api()

Returns NotImplementedError - needs to be implemented in subclass

id

Returns int with the ID

class rtkit.entities.**Links**(*id*)
Bases: rtkit.entities.RTEntity

Links Object

Todo

Links not implemented

---

static api()

Returns NotImplementedError - needs to be implemented in subclass

id

Returns int with the ID

2.4 rtkit.errors

exception rtkit.errors.**RTUnknownTypeError**(*msg=None*, *http_code=None*, *response=None*)
Bases: rtkit.errors.RTResourceError

Unknown Type Exception

args

message

status_int = 400

exception rtkit.errors.**RTInvalidError**(*msg=None*, *http_code=None*, *response=None*)
Bases: rtkit.errors.RTResourceError

Invalid Exception

args

message

status_int = 400

exception rtkit.errors.**RTValueError**(*msg=None*, *http_code=None*, *response=None*)
Bases: rtkit.errors.RTResourceError

value Error Exception

args

message

status_int = 400

exception rtkit.errors.**RTResourceError**(*msg=None*, *http_code=None*, *response=None*)
Bases: exceptions.Exception

Default error class

args

message

status_int = None

exception rtkit.errors.**RTNotFoundError**(*msg=None*, *http_code=None*, *response=None*)
Bases: rtkit.errors.RTResourceError

Not Found Exception

args

message

status_int = 404

exception rtkit.errors.**RTUnauthorized**(*msg=None*, *http_code=None*, *response=None*)
Bases: rtkit.errors.RTResourceError

Not Authorised Exception

args

message

status_int = 401

2.5 rst.forms

class rtkit.forms.**BoundaryItem**(*name*, *value*, *fname=None*, *filetype=None*, *filesize=None*)
Bases: object

encode(*boundary*)

Returns A string encoding of this parameter.

encode_hdr(*boundary*)

Returns The header of the encoding of this parameter

encode_unreadable_value(*value*)

iter_encode(*boundary*, *blocksize=16384*)

class rtkit.forms.**MultipartForm**(*params*, *boundary*)
Bases: object

Represents an encoded Form

get_size()

rtkit.forms.**encode**(*value*, *headers*)

rtkit.forms.**to_bytestring**(*s*)

rtkit.forms.**url_quote**(*s*, *charset='utf-8'*, *safe='/:'*)
URL encode a single string with a given encoding.

2.6 rtkit.parser

class rtkit.parser.**RTParser**

Bases: object

RFC5322 Parser - see https://tools.ietf.org/html/rfc5322

COMMENT = <_sre.SRE_Pattern object at 0x38bc1c0>

HEADER = <_sre.SRE_Pattern object at 0x390bc40>

SECTION = <_sre.SRE_Pattern object at 0x36f43e8>

classmethod build(*body*)

Build logical lines from a RFC5322-like string

Returns A list of strings

```
>>> body = '''RT/1.2.3 200 Ok
...
... # a
...   b
... spam: 1
...
... ham: 2,
...      3
... --
... # c
... spam: 4
... ham:
... --
... a -- b
... '''
>>> RTParser.build(body)
[['# a b', 'spam: 1', 'ham: 2, 3'], ['# c', 'spam: 4', 'ham:'], ['a -- b']]
```

classmethod decode(*lines*)

Returns A list of 2-tuples parsing 'k: v' and skipping comments

```
>>> RTParser.decode(['# c1: c2', 'spam: 1', 'ham: 2, 3', 'eggs:'])
[('spam', '1'), ('ham', '2, 3'), ('eggs', '')]
>>> RTParser.decode(['<!DOCTYPE HTML PUBLIC >', '<html><head>',])
[]
```

classmethod decode_comment(*lines*)

Returns A list of 2-tuples parsing '# k: v'

```
>>> RTParser.decode_comment(['# c1: c2', 'spam: 1', 'ham: 2, 3', 'eggs:'])
[('c1', 'c2')]
>>>
```

classmethod parse(*body*, *decoder*)

Returns A list of RFC5322-like section

```
>>> decode = RTParser.decode
>>> body = '''
...
... # c1
... spam: 1
```

```
... ham: 2,
...      3
... eggs:'''
>>> RTParser.parse(body, decode)
[[('spam', '1'), ('ham', '2, 3'), ('eggs', '')]]
>>> RTParser.parse('# spam 1 does not exist.', decode)
Traceback (most recent call last):
...
RTNotFoundError: spam 1 does not exist
>>> RTParser.parse('# Spam 1 created.', decode)
[[('id', 'spam/1')]]
>>> RTParser.parse('No matching results.', decode)
[]
>>> decode = RTParser.decode_comment
>>> RTParser.parse('# spam: 1\n# ham: 2', decode)
[[('spam', '1'), ('ham', '2')]]
```

2.7 rtkit.resource

class `rtkit.resource.`**`RTObj`**(*dic=None*)

RT Simple Object Container. This is a quick hack to make the data returned slightly oo with some helpers.

Parameters **dic** – The dictionary to make Oo

```
mydata = {'Foo': 'Var'}
myObj = RTObj(mydata)
print myObj.Foo

resource = RTResource('http://rt.example.com/REST/1.0/', 'webmaster', 'secret', CookieAuthentica
try:
    response = resource.get(path='ticket/28')
    myTicket = response.as_object() ## Returns an RtObj instance

except RTResourceError as e:
    logger.error(e.response.status_int)
    logger.error(e.response.status)
    logger.error(e.response.parsed)

## Show Stuff
print myTicket.Subject, myTicket.id
print myTicket.get_custom("my_custom")
print myTicket.keys()  # list of keys
print mtTicket.as_dict() # return as dict and key/value pair

## Update Stuff
myTicket.set_custom("my_custom", "New Val")
myTicker.Subject = myTicket.Subject + " my Xtra"
```

`as_dict`()

Returns Dictionary of data as key value pairs

`get`(*name*)

This is useful as some "keys" are not ooable eg ob.get("X-yx");

Parameters **name** – The value to get

Returns The value of name.

get_custom(*name*)
Get a custom field, a short cut to CF.{name}

Parameters **name** – of custom field eg 'Works Order'

Returns the data as string

keys()

Returns A list with strings of the field names

set_custom(*name*, *val*)
Set a custom field, a short cut to CF.{name}

Parameters

- **name** – of custom field eg 'Works Order'
- **val** – New value to set to

class rtkit.resource.**RTResource**(*url*, *username*, *password*, *auth*, ***kwargs*)
Bases: object

REST Resource Object

Create Connection Object

Parameters

- **url** – Server URL
- **username** – RT Login
- **password** – Password
- **auth** – Instance of rtkit.authenticators

get(*path=None*, *headers=None*)
GET from the server

post(*path=None*, *payload=None*, *headers=None*)
POST to the server

request(*method*, *path=None*, *payload=None*, *headers=None*)
Make request to server

class rtkit.resource.**RTResponse**(*request*, *response*)
Bases: object

Represents the REST response from server

as_dict()

Returns dict with the data

as_object()

Returns A rtkit.resource.RTObj object instance

body = None
Request Body

headers = None
Headers as dict

logger = None
Logger

parsed = None
A List of Tuples of data

status = None
Status String

status_int = None
Status Code

2.8 rtkit.tracker

class rtkit.tracker.**Tracker**(*url*, *username*, *password*, *auth*, *language='en'*)
Bases: rtkit.resource.RTResource

Tracker Object

change_links(*ticket_id*, *content*)
Change Links

> **Warning:** Not yet Implemented

comment_ticket(*content*, *attachments=None*)
Comment on Ticket

> **Warning:** Not yet Implemented

create_ticket(*content*, *attachments=None*)
Create a ticket

> **Warning:** Not yet Implemented

get(*path=None*, *headers=None*)
GET from the server

get_attachment(*ticket_id*, *value*)

Returns An instance of rtkit.entities.Attachment

get_history(*ticket_id*, *value=None*, *format='l'*)

Returns An instance of rtkit.entities.History

get_links(*ticket_id*)

Returns An instance of rtkit.entities.Links

get_queue(*value*)

Returns An instance of rtkit.entities.Queue

get_ticket(*value*)

Returns An instance of rtkit.entities.Ticket

get_user(*value*)

Returns An instance of rtkit.entities.User

post(*path=None*, *payload=None*, *headers=None*)
POST to the server

request (*method*, *path=None*, *payload=None*, *headers=None*)
Make request to server

search_tickets (*query*, *order*)
Search tickets

> **Warning:** Not yet Implemented

2.9 Developers Guide

Once upon a time, z4r created a pushed some code to github; little did he know what was to happen next, when the fork.. .. developers guide to be written.. continued.....

2.9.1 TODO List

(extracted from source)

Todo

History not implemented

(The *original entry* is located in /var/build/user_builds/rtkit/checkouts/latest/rtkit/entities.py:docstring of rtkit.entities.History, line 3.)

Todo

Links not implemented

(The *original entry* is located in /var/build/user_builds/rtkit/checkouts/latest/rtkit/entities.py:docstring of rtkit.entities.Links, line 3.)

CHAPTER

THREE

CREDITS

abstract Python interface to Request Tracker REST API

version 0.3.1

author Andrea De Marco <24erre@gmail.com>

contact http://z4r.github.com/

date 2011-07-15

3.1 Indices and tables

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

PYTHON MODULE INDEX

r