
python-rtkit Documentation

Release

Andrea De Marco <24erre@gmail.com>

February 24, 2017

1	Request Tracker REST Interface	1
1.1	Installation	1
1.2	RT REST API Summary	1
1.3	Authentication Handlers	2
1.4	Overview on Low Level API	3
1.5	License	6
1.6	References	6
2	rtkit	7
2.1	rtkit.authenticators	7
2.2	rtkit.comment	9
2.3	rtkit.entities	10
2.4	rtkit.errors	13
2.5	rst.forms	14
2.6	rtkit.parser	14
2.7	rtkit.resource	16
2.8	rtkit.tracker	16
2.9	Developers Guide	17
3	Credits	19
3.1	Indices and tables	19
	Python Module Index	21

Request Tracker REST Interface

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

- *Installation*
- *RT REST API Summary*
- *Authentication Handlers*
 - *Basic Authentication*
 - *Cookie-based Authentication*
 - *QueryString Authentication*
 - *Kerberos Authentication*
- *Overview on Low Level API*
 - *Create ticket*
 - *Read a ticket*
 - *Edit a ticket or ticket's links*
 - *Comment on a Ticket with Attachments*
 - *Custom Fields*
- *License*
- *References*

Installation

Using pip:

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

Using pip dev:

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

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>

Authentication Handlers

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)
```

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)
```

QueryString Authentication

```
from rtkit.resource import RTResource
from rtkit.authenticators import QueryStringAuthenticator
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>', QueryStringAuthenticator)
```

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: Remember to install `urllib2_kerberos`.

Overview on Low Level API

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 queue
[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] []
```

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] []
```

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': 'pyRTkit/0.0.1'}
[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\n# spam: Unknown field.\n\n\n'
[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.']]
```

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)
```

Custom Fields

To create or update a tkt with Custom Fields you must use this notation:

```
content = {
    'content': {
        'Queue': 1,
        'Subject' : 'New Ticket',
        'Text' : 'My useless\ntext on\nthree lines.',
        'CF.{Need For Approval}': 'Yes'
    }
}
```

Warning: With RT/3.8 you can't use ? inside the names of your custom fields, with RT/3.6 / () too.

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.

References

- [Best Practical RT](#)
- [Request Tracker Wiki](#)

rtkit.authenticators

Connect to an RT server using various authentication techniques

- The *AbstractAuthenticator* contains the base methods
- **And the current implementations are:**
 - *BasicAuthenticator*
 - *CookieAuthenticator*
 - *QueryStringAuthenticator*
 - *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.**QueryStringAuthenticator**(username, password, url)
Bases: *rtkit.authenticators.AbstractAuthenticator*
Authenticate against server using a querystring

```
from rtkit.resource import RTResource
from rtkit.authenticators import QueryStringAuthenticator
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>', QueryStringAuthenticator)
```

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

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
```

rtkit.entities

class rtkit.entities.**RTEntity**(*id*, *tracker*)

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*, *tracker*, ***kwargs*)

Bases: *rtkit.entities.RTEntity*

User Object

static api()

Returns str with 'user'

id

Returns int with the ID

```

class rtkit.entities.Queue(id, tracker, **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

    search_tickets (query='', active=True, order='id')

class rtkit.entities.Ticket(id, tracker, **kwargs)
    Bases: rtkit.entities.RTEntity
    Ticket Object
    static api ()
        Returns str with 'ticket'

    creator = None
        Creator

    date = None
        Dates as dictionary 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

    requestors = None
        Requestors

```

status = None

Status

subject = None

Subject

class rtkit.entities.**Attachment** (*id, tracker, **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, tracker*)

Bases: *rtkit.entities.RTEntity*

History Object

Todo

History not implemented

api ()

Returns NotImplementedError - needs to be implemented in subclass

id

Returns int with the ID

class rtkit.entities.**Links** (*id, tracker*)

Bases: *rtkit.entities.RTEntity*

Links Object

Todo

Links not implemented

api ()

Returns NotImplementedError - needs to be implemented in subclass

id

Returns int with the ID

rtkit.errors

exception `rtkit.errors.RTBadConfiguration`

Bases: `exceptions.Exception`

args

message

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
```

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.
```

rtkit.parser

```
class rtkit.parser.RTParser
    Bases: object
```

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

```
COMMENT = <_sre.SRE_Pattern object>
```

```
HEADER = <_sre.SRE_Pattern object>
```

```
SECTION = <_sre.SRE_Pattern object>
```

```
SYNTAX_COMMENT = <_sre.SRE_Pattern object>
```

```
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)
[[u'# a\nb', u'spam: 1', u'ham: 2,\n3'], [u'# c', u'spam: 4', u'ham:'], [u'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')]
>>> RTParser.decode_comment(['# Syntax error.', '>> c1: c2', '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,\n3'), ('eggs', '')]]
>>> RTParser.parse('# spam 1 does not exist.', decode)
Traceback (most recent call last):
...
RTNotNotFoundError: 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')]]

```

rtkit.resource

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`

classmethod `from_rtrc` (*auth, filename=None, **kwargs*)

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

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

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

from_rtrc (*auth, filename=None, **kwargs*)

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=None*)

Search tickets :return: A list of `rtkit.entities.Ticket` instances

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.....

TODO List

(extracted from source)

Todo

History not implemented

(The original entry is located in `/home/docs/checkouts/readthedocs.org/user_builds/python-rtkit/checkouts/stable/rtkit/entities.py:docstring of rtkit.entities.History`, line 3.)

Todo

Links not implemented

(The original entry is located in `/home/docs/checkouts/readthedocs.org/user_builds/python-rtkit/checkouts/stable/rtkit/entities.py:docstring of rtkit.entities.Links`, line 3.)

Credits

abstract Python interface to Request Tracker REST API

version 0.7.1

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

contact <http://z4r.github.com/>

date 2011-07-15

copyright Copyright (C) 2011, Andrea De Marco <24erre@gmail.com>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

Indices and tables

- [genindex](#)
- [modindex](#)
- [search](#)

r

`rtkit`, [19](#)
`rtkit.authenticators`, [7](#)
`rtkit.comment`, [9](#)
`rtkit.entities`, [10](#)
`rtkit.errors`, [13](#)
`rtkit.forms`, [14](#)
`rtkit.parser`, [14](#)
`rtkit.resource`, [16](#)
`rtkit.tracker`, [16](#)

A

AbstractAuthenticator (class in rtkit.authenticators), 7
api() (rtkit.entities.Attachment static method), 12
api() (rtkit.entities.History method), 12
api() (rtkit.entities.Links method), 12
api() (rtkit.entities.Queue static method), 11
api() (rtkit.entities.RTEntity static method), 10
api() (rtkit.entities.Ticket static method), 11
api() (rtkit.entities.User static method), 10
args (rtkit.comment.RTCreated attribute), 9
args (rtkit.comment.RTNoMatch attribute), 9
args (rtkit.errors.RTBadConfiguration attribute), 13
args (rtkit.errors.RTInvalidError attribute), 13
args (rtkit.errors.RTNotFoundError attribute), 13
args (rtkit.errors.RTResourceError attribute), 13
args (rtkit.errors.RTUnauthorized attribute), 13
args (rtkit.errors.RTUnknownTypeError attribute), 13
args (rtkit.errors.RTValueError attribute), 13
Attachment (class in rtkit.entities), 12

B

BasicAuthenticator (class in rtkit.authenticators), 7
body (rtkit.resource.RTResponse attribute), 16
BoundaryItem (class in rtkit.forms), 14
build() (rtkit.parser.RTParser class method), 14

C

change_links() (rtkit.tracker.Tracker method), 16
check() (in module rtkit.comment), 9
COMMENT (rtkit.parser.RTParser attribute), 14
comment_ticket() (rtkit.tracker.Tracker method), 17
content (rtkit.entities.Attachment attribute), 12
CookieAuthenticator (class in rtkit.authenticators), 8
create_ticket() (rtkit.tracker.Tracker method), 17
creator (rtkit.entities.Ticket attribute), 11
ctype (rtkit.entities.Attachment attribute), 12

D

date (rtkit.entities.Ticket attribute), 11
decode() (rtkit.parser.RTParser class method), 15

decode_comment() (rtkit.parser.RTParser class method), 15
delta (rtkit.entities.Ticket attribute), 11
description (rtkit.entities.Queue attribute), 11

E

encode() (in module rtkit.forms), 14
encode() (rtkit.forms.BoundaryItem method), 14
encode_hdr() (rtkit.forms.BoundaryItem method), 14
encode_unreadable_value() (rtkit.forms.BoundaryItem method), 14
encoding (rtkit.entities.Attachment attribute), 12

F

filename (rtkit.entities.Attachment attribute), 12
from_rtrc() (rtkit.resource.RTResource class method), 16
from_rtrc() (rtkit.tracker.Tracker method), 17

G

get() (rtkit.resource.RTResource method), 16
get() (rtkit.tracker.Tracker method), 17
get_attachment() (rtkit.tracker.Tracker method), 17
get_history() (rtkit.tracker.Tracker method), 17
get_links() (rtkit.tracker.Tracker method), 17
get_queue() (rtkit.tracker.Tracker method), 17
get_size() (rtkit.forms.MultipartForm method), 14
get_ticket() (rtkit.tracker.Tracker method), 17
get_user() (rtkit.tracker.Tracker method), 17

H

HEADER (rtkit.parser.RTParser attribute), 14
headers (rtkit.resource.RTResponse attribute), 16
History (class in rtkit.entities), 12

I

id (rtkit.entities.Attachment attribute), 12
id (rtkit.entities.History attribute), 12
id (rtkit.entities.Links attribute), 12
id (rtkit.entities.Queue attribute), 11
id (rtkit.entities.RTEntity attribute), 10

id (rtkit.entities.Ticket attribute), 11
 id (rtkit.entities.User attribute), 10
 iter_encode() (rtkit.forms.BoundaryItem method), 14

K

KerberosAuthenticator (class in rtkit.authenticators), 8

L

Links (class in rtkit.entities), 12
 logger (rtkit.resource.RTResponse attribute), 16
 login() (rtkit.authenticators.AbstractAuthenticator method), 7
 login() (rtkit.authenticators.BasicAuthenticator method), 8
 login() (rtkit.authenticators.CookieAuthenticator method), 8
 login() (rtkit.authenticators.KerberosAuthenticator method), 9
 login() (rtkit.authenticators.QueryStringAuthenticator method), 8

M

message (rtkit.comment.RTCreated attribute), 9
 message (rtkit.comment.RTNoMatch attribute), 9
 message (rtkit.errors.RTBadConfiguration attribute), 13
 message (rtkit.errors.RTInvalidError attribute), 13
 message (rtkit.errors.RTNotFoundError attribute), 13
 message (rtkit.errors.RTResourceError attribute), 13
 message (rtkit.errors.RTUnauthorized attribute), 14
 message (rtkit.errors.RTUnknownTypeError attribute), 13
 message (rtkit.errors.RTValueError attribute), 13
 MultipartForm (class in rtkit.forms), 14

N

name (rtkit.entities.Queue attribute), 11

O

open() (rtkit.authenticators.AbstractAuthenticator method), 7
 open() (rtkit.authenticators.BasicAuthenticator method), 8
 open() (rtkit.authenticators.CookieAuthenticator method), 8
 open() (rtkit.authenticators.KerberosAuthenticator method), 9
 open() (rtkit.authenticators.QueryStringAuthenticator method), 8
 owner (rtkit.entities.Ticket attribute), 11

P

parse() (rtkit.parser.RTParser class method), 15
 parsed (rtkit.resource.RTResponse attribute), 16
 post() (rtkit.resource.RTResource method), 16

post() (rtkit.tracker.Tracker method), 17
 priority (rtkit.entities.Ticket attribute), 11

Q

QueryStringAuthenticator (class in rtkit.authenticators), 8
 Queue (class in rtkit.entities), 10

R

request() (rtkit.resource.RTResource method), 16
 request() (rtkit.tracker.Tracker method), 17
 requestors (rtkit.entities.Ticket attribute), 11
 RTBadConfiguration, 13
 RTCreated, 9
 RTEntity (class in rtkit.entities), 10
 RTInvalidError, 13
 rtkit (module), 19
 rtkit.authenticators (module), 7
 rtkit.comment (module), 9
 rtkit.entities (module), 10
 rtkit.errors (module), 13
 rtkit.forms (module), 14
 rtkit.parser (module), 14
 rtkit.resource (module), 16
 rtkit.tracker (module), 16
 RTNoMatch, 9
 RTNotFoundError, 13
 RTParser (class in rtkit.parser), 14
 RTResource (class in rtkit.resource), 16
 RTResourceError, 13
 RTResponse (class in rtkit.resource), 16
 RTUnauthorized, 13
 RTUnknownTypeError, 13
 RTValueError, 13

S

search_tickets() (rtkit.entities.Queue method), 11
 search_tickets() (rtkit.tracker.Tracker method), 17
 SECTION (rtkit.parser.RTParser attribute), 14
 status (rtkit.entities.Ticket attribute), 11
 status (rtkit.resource.RTResponse attribute), 16
 status_int (rtkit.errors.RTInvalidError attribute), 13
 status_int (rtkit.errors.RTNotFoundError attribute), 13
 status_int (rtkit.errors.RTResourceError attribute), 13
 status_int (rtkit.errors.RTUnauthorized attribute), 14
 status_int (rtkit.errors.RTUnknownTypeError attribute), 13
 status_int (rtkit.errors.RTValueError attribute), 13
 status_int (rtkit.resource.RTResponse attribute), 16
 subject (rtkit.entities.Ticket attribute), 12
 SYNTAX_COMMENT (rtkit.parser.RTParser attribute), 14

T

Ticket (class in rtkit.entities), 11

`to_bytestring()` (in module `rtkit.forms`), [14](#)
`Tracker` (class in `rtkit.tracker`), [16](#)

U

`url_quote()` (in module `rtkit.forms`), [14](#)
`User` (class in `rtkit.entities`), [10](#)