f5-icontrol-rest Documentation

Release 1.0.9

F5 Networks

August 03, 2016

Contents

CHAPTER 1

Overview

The F5 Networks® *icontrol* module is used to send commands to the BIGIP® iControl® REST API. The library maintains a HTTP session (which is a `requests.Session`) and does URL validation and logging.

CHAPTER 2

Installation

2.1 Using Pip

```
$ pip install f5-icontrol-rest
```

2.2 GitHub

F5Networks/f5-icontrol-rest-python

CHAPTER 3

Examples

```
from icontrol.session import iControlRESTSession
icr_session = iControlRESTSession('myuser', 'mypass')

# GET to https://bigip.example.com/mgmt/tm/ltm/nat/~Common~mynat
icr_session.get(
    'https://bigip.example.com/mgmt/tm/ltm/nat',
    name='mynat',
    partition='Common')

# GET to https://bigip.example.com/mgmt/tm/ltm/nat
icr_session.get('https://bigip.example.com/mgmt/tm/ltm/nat')

# POST with json data
icr_session.post('https://bigip.example.com/mgmt/tm/ltm/nat', \
json={'name': 'myname', 'partition': 'Common'})
```

CHAPTER 4

Module Documentation

4.1 icontrol

4.1.1 icontrol package

Submodules

icontrol.authtoken module

A requests-compatible system for BIG-IP token-based authentication.

BIG-IP only allows users with the Administrator role to authenticate to iControl using HTTP Basic auth. Non-Administrator users can use the token-based authentication scheme described at:

https://devcentral.f5.com/wiki/icontrol.authentication_with_the_f5_rest_api.ashx

Use this module with requests to automatically get a new token, and attach `requests.Session` object, so that it is used to authenticate future requests.

Instead of using this module directly, it is easiest to enable it by passing a `token=True` argument when creating the *iControlRESTSession*:

```
>>> iCRS = iControlRESTSession('bob', 'secret', token=True)
```

class icontrol.authtoken.**iControlRESTTokenAuth**(*username*, *password*, *login_provider_name='tmos'*)

Bases: `requests.auth.AuthBase`

Acquire and renew BigIP iControl REST authentication tokens.

Parameters

- **username** (*str*) – The username on BigIP
- **password** (*str*) – The password for username on BigIP
- **login_provider_name** (*str*) – The name of the login provider that BigIP should consult when creating the token.

If `username` is configured locally on the BigIP, `login_provider_name` should be `"tmos"` (default). Otherwise (for example, `username` is configured on LDAP that BigIP consults), consult BigIP documentation or your system administrator for the value of `login_provider_name`.

get_new_token(*netloc*)

Get a new token from BIG-IP and store it internally.

Throws relevant exception if it fails to get a new token.

This method will be called automatically if a request is attempted but there is no authentication token, or the authentication token is expired. It is usually not necessary for users to call it, but it can be called if it is known that the authentication token has been invalidated by other means.

icontrol.exceptions module

Exceptions that can be emitted by the icontrol package.

exception icontrol.exceptions.**BigIPInvalidURL**
Bases: exceptions.Exception

exception icontrol.exceptions.**InvalidBigIP_ICRURI**
Bases: *icontrol.exceptions.BigIPInvalidURL*

exception icontrol.exceptions.**InvalidInstanceNameOrFolder**
Bases: *icontrol.exceptions.BigIPInvalidURL*

exception icontrol.exceptions.**InvalidPrefixCollection**
Bases: *icontrol.exceptions.BigIPInvalidURL*

exception icontrol.exceptions.**InvalidScheme**
Bases: *icontrol.exceptions.BigIPInvalidURL*

exception icontrol.exceptions.**InvalidSuffixCollection**
Bases: *icontrol.exceptions.BigIPInvalidURL*

exception icontrol.exceptions.**iControlUnexpectedHTTPError**(**args*, ***kwargs*)
Bases: requests.exceptions.HTTPError

icontrol.session module

A BigIP-RESTServer URI handler. REST-APIs use it on the `requests` library.

Use this module to make calls to a BigIP-REST server. It will handle:

1. URI Sanitization uri's produced by this module are checked to ensure compliance with the BigIP-REST server interface
2. Session Construction – the *iControlRESTSession* wraps a `requests.Session` object.
3. Logging – pre- and post- request state is logged.
4. Exception generation – Errors in URL construction generate `BigIPInvalidURL` subclasses; unexpected HTTP status codes raise `iControlUnexpectedHTTPError`.

The core functionality of the module is implemented via the *iControlRESTSession* class. Calls to its' HTTP-methods are checked, pre-logged, submitted, and post-logged.

There are 2 modes of operation "full_uri", and "uri_as_parts", toggled by the *uri_as_parts* boolean keyword param that can be passed to methods. It defaults to *False*. Use *uri_as_parts* when you want to leverage the full functionality of this library, and have it construct your uri for you. Example Use in *uri_as_parts* mode:

```
>>> iCRS = iControlRESTSession('jrandomhacker', 'insecure')
>>> iCRS.get('https://192.168.1.1/mgmt/tm/ltm/nat/', partition='Common', name='VALIDNAME', uri_as_pa
```

In *full_uri* mode:

```
>>> iCRS.get('https://192.168.1.1/mgmt/tm/ltm/nat/~Common~VALIDNAME')
```

NOTE: If used via the `f5-common-python` library the typical mode is "full_uri" since that library binds uris to Python objects.

Available functions:

- iCRS.{get, post, put, delete, patch}: requests.Session.VERB wrappers
- decorate_HTTP_verb_method: this function preps, logs, and handles requests

against the BigIP REST Server, by pre- and post- processing the above methods.

`icontrol.session.`**`decorate_HTTP_verb_method`**(*method*)

Prepare and Post-Process HTTP VERB method for BigIP-RESTServer request.

This function decorates all of the HTTP VERB methods in the iControlRESTSession class. It provides the core logic for this module. If necessary it validates and assembles a uri from parts with a call to *generate_bigip_uri*.

Then it:

1.pre-logs the details of the request

2.submits the request

3.logs the response, included expected status codes

4.raises exceptions for unexpected status codes. (i.e. not doc'd as BigIP RESTServer codes.)

`icontrol.session.`**`generate_bigip_uri`**(*base_uri*, *partition*, *name*, *suffix*, ***kwargs*)

(str, str, str) –> str

This function checks the supplied elements to see if each conforms to the specifiction for the appropriate part of the URI. These validations are conducted by the helper function _validate_uri_parts. After validation the parts are assembled into a valid BigIP REST URI string which is then submitted with appropriate metadata.

```
>>> generate_bigip_uri('https://0.0.0.0/mgmt/tm/ltm/nat/',     'CUSTOMER1', 'nat52', params={'a'
'https://0.0.0.0/mgmt/tm/ltm/nat/~CUSTOMER1~nat52'
>>> generate_bigip_uri('https://0.0.0.0/mgmt/tm/ltm/nat/',     'CUSTOMER1', 'nat52', params={'a'
'https://0.0.0.0/mgmt/tm/ltm/nat/~CUSTOMER1~nat52/wacky'
>>> generate_bigip_uri('https://0.0.0.0/mgmt/tm/ltm/nat/', '', '',     params={'a':1}, suffix='/
'https://0.0.0.0/mgmt/tm/ltm/nat/thwocky'
```

class `icontrol.session.`**`iControlRESTSession`**(*username*, *password*, ***kwargs*)

Bases: `object`

Represents a `requests.Session` that communicates with a BigIP.

Instantiate one of these when you want to communicate with a BigIP-REST Server, it will handle BigIP-specific details of the uri's. In the f5-common-python library, an *`iControlRESTSession`* is instantiated during BigIP instantiation and associated with it as an attribute of the BigIP (a compositional vs. inheritable association).

Objects instantiated from this class provide an HTTP 1.1 style session, via the `requests.Session` object, and HTTP-methods that are specialized to the BigIP-RESTServer interface.

Pass `token=True` in `**kwargs` to use token-based authentication. This is required for users that do not have the Administrator role on BigIP.

`delete`(*RIC_base_uri*, ***kwargs*)

Sends a HTTP DELETE command to the BIGIP REST Server.

Use this method to send a DELETE command to the BIGIP. When calling this method with the optional arguments `name` and `partition` as part of `**kwargs` they will be added to the `uri` passed in separated by ~ to create a proper BIGIP REST API URL for objects.

All other parameters passed in as `**kwargs` are passed directly to the `requests.Session.delete()`

Parameters

- **uri** (*str*) – A HTTP URI
- **name** (*str*) – The object name that will be appended to the uri
- **partition** (*str*) – The partition name that will be appened to the uri
- ****kwargs** – The `reqeusts.Session.delete()` optional params

get (*RIC_base_uri*, ***kwargs*)
Sends a HTTP GET command to the BIGIP REST Server.

Use this method to send a GET command to the BIGIP. When calling this method with the optional arguments `name` and `partition` as part of `**kwargs` they will be added to the `uri` passed in separated by ~ to create a proper BIGIP REST API URL for objects.

All other parameters passed in as `**kwargs` are passed directly to the `requests.Session.get()`

Parameters

- **uri** (*str*) – A HTTP URI
- **name** (*str*) – The object name that will be appended to the uri
- **partition** (*str*) – The partition name that will be appened to the uri
- ****kwargs** – The `reqeusts.Session.get()` optional params

patch (*RIC_base_uri*, ***kwargs*)
Sends a HTTP PATCH command to the BIGIP REST Server.

Use this method to send a PATCH command to the BIGIP. When calling this method with the optional arguments `name` and `partition` as part of `**kwargs` they will be added to the `uri` passed in separated by ~ to create a proper BIGIP REST API URL for objects.

All other parameters passed in as `**kwargs` are passed directly to the `requests.Session.patch()`

Parameters

- **uri** (*str*) – A HTTP URI
- **data** (*str*) – The data to be sent with the PATCH command
- **name** (*str*) – The object name that will be appended to the uri
- **partition** (*str*) – The partition name that will be appened to the uri
- ****kwargs** – The `reqeusts.Session.patch()` optional params

post (*RIC_base_uri*, ***kwargs*)
Sends a HTTP POST command to the BIGIP REST Server.

Use this method to send a POST command to the BIGIP. When calling this method with the optional arguments `name` and `partition` as part of `**kwargs` they will be added to the `uri` passed in separated by ~ to create a proper BIGIP REST API URL for objects.

All other parameters passed in as `**kwargs` are passed directly to the `requests.Session.post()`

Parameters

- **uri** (*str*) – A HTTP URI
- **data** (*str*) – The data to be sent with the POST command

- **json** (*dict*) – The JSON data to be sent with the POST command
- **name** (*str*) – The object name that will be appended to the uri
- **partition** (*str*) – The partition name that will be appened to the uri
- ****kwargs** – The `reqeusts.Session.post()` optional params

put(*RIC_base_uri*, ***kwargs*)

Sends a HTTP PUT command to the BIGIP REST Server.

Use this method to send a PUT command to the BIGIP. When calling this method with the optional arguments `name` and `partition` as part of `**kwargs` they will be added to the `uri` passed in separated by ~ to create a proper BIGIP REST API URL for objects.

All other parameters passed in as `**kwargs` are passed directly to the `requests.Session.put()`

Parameters

- **uri** (*str*) – A HTTP URI
- **data** (*str*) – The data to be sent with the PUT command
- **json** (*dict*) – The JSON data to be sent with the PUT command
- **name** (*str*) – The object name that will be appended to the uri
- **partition** (*str*) – The partition name that will be appened to the uri
- ****kwargs** – The `reqeusts.Session.put()` optional params

Module contents

CHAPTER 5

License

5.1 Apache V2.0

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

5.2 Contributor License Agreement

Individuals or business entities who contribute to this project must have completed and submitted the F5® Contributor License Agreement to Openstack_CLA@f5.com prior to their code submission being included in this project.

i

B

D

G

I

P