

---

# **f5-icontrol-rest Documentation**

***Release 1.3.10***

**F5 Networks**

**Aug 04, 2018**



---

## Contents

---

<b>1</b>	<b>Overview</b>	<b>1</b>
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	Using Pip . . . . .	3
2.2	GitHub . . . . .	3
<b>3</b>	<b>Examples</b>	<b>5</b>
<b>4</b>	<b>Module Documentation</b>	<b>7</b>
4.1	icontrol . . . . .	7
4.1.1	icontrol package . . . . .	7
4.1.1.1	Submodules . . . . .	7
4.1.1.2	icontrol.authtoken module . . . . .	7
4.1.1.3	icontrol.exceptions module . . . . .	8
4.1.1.4	icontrol.session module . . . . .	8
4.1.1.5	Module contents . . . . .	12
<b>5</b>	<b>License</b>	<b>13</b>
5.1	Apache V2.0 . . . . .	13
5.2	Contributor License Agreement . . . . .	13
	<b>Python Module Index</b>	<b>15</b>



# 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.



#### 2.1 Using Pip

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

#### 2.2 GitHub

[F5Networks/f5-icontrol-rest-python](#)





## CHAPTER 3

---

### Examples

---

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

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

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

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



## 4.1 iconcontrol

### 4.1.1 iconcontrol package

#### 4.1.1.1 Submodules

#### 4.1.1.2 iconcontrol.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/iconcontrol.authentication\\_with\\_the\\_f5\\_rest\\_api.ashx](https://devcentral.f5.com/wiki/iconcontrol.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 iconcontrol.authtoken.iControlRESTTokenAuth(username, password, login_provider_name='tmos', verify=False, auth_provider=None)
```

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.
- **verify** (*str*) – The path to a CA bundle containing the CA certificate for SSL validation

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\_auth\_providers** (*netloc*)

BIG-IQ specific query for auth providers

BIG-IP doesn't really need this because BIG-IP's multiple auth providers seem to handle fallback just fine. BIG-IQ on the other hand, needs to have its auth provider specified if you're using one of the non-default ones.

**Parameters** *netloc* –

**Returns**

#### **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.

### 4.1.1.3 **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.InvalidURIComponentPart`

Bases: `icontrol.exceptions.BigIPInvalidURL`

**exception** `icontrol.exceptions.iControlUnexpectedHTTPError` (*\*args*, *\*\*kwargs*)

Bases: `requests.exceptions.HTTPError`

### 4.1.1.4 **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/ltn/nat/', partition='Common', name=
↳ 'VALIDNAME', uri_as_parts=True)
```

In `full_uri` mode:

```
>>> iCRS.get('https://192.168.1.1/mgmt/tm/ltn/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.debug_prepared_request(request)`

`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, sub_path, suffix, **kwargs)`  
(str, str, str) -> str

This function checks the supplied elements to see if each conforms to the specification 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/ltn/nat/', 'CUSTOMER1', 'nat52'
↳ ', params={'a':1})
'https://0.0.0.0/mgmt/tm/ltn/nat/~CUSTOMER1~nat52'
```

(continues on next page)

(continued from previous page)

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

**::Warning:** There are cases where `'/'` and `'~'` characters are valid in the object name. This is indicated by passing `'transform_name'` boolean as `True`, by default this is set to `False`.

`icontrol.session.get_request_args(kwargs)`

`icontrol.session.get_send_args(kwargs)`

**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.

**append\_user\_agent** (`user_agent`)

Append text to the User-Agent header for the request.

Use this method to update the User-Agent header by appending the given string to the session's User-Agent header separated by a space.

**Parameters** `user_agent` (`str`) – A string to append to the User-Agent header

**debug**

**debug\_output**

**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 appended to the uri
- **\*\*kwargs** – The `requests.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 appended to the uri
- **\*\*kwargs** – The `requests.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 appended to the uri
- **\*\*kwargs** – The `requests.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 appended to the uri
- **\*\*kwargs** – The `requests.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 appended to the uri
- **\*\*kwargs** – The `requests.Session.put()` optional params

#### token

Convenience wrapper around returning the current token

**Returns:** result (str): The current token being sent in session headers.

#### 4.1.1.5 Module contents



### 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](mailto:Openstack_CLA@f5.com) prior to their code submission being included in this project.



### i

- `icontrol`, [12](#)
- `icontrol.authtoken`, [7](#)
- `icontrol.exceptions`, [8](#)
- `icontrol.session`, [8](#)



## A

append\_user\_agent() (icontrol.session.iControlRESTSession method), 10

## B

BigIPInvalidURL, 8

## D

debug (icontrol.session.iControlRESTSession attribute), 10

debug\_output (icontrol.session.iControlRESTSession attribute), 10

debug\_prepared\_request() (in module icontrol.session), 9

decorate\_HTTP\_verb\_method() (in module icontrol.session), 9

delete() (icontrol.session.iControlRESTSession method), 10

## G

generate\_bigip\_uri() (in module icontrol.session), 9

get() (icontrol.session.iControlRESTSession method), 10

get\_auth\_providers() (icontrol.authtoken.iControlRESTTokenAuth method), 8

get\_new\_token() (icontrol.authtoken.iControlRESTTokenAuth method), 8

get\_request\_args() (in module icontrol.session), 10

get\_send\_args() (in module icontrol.session), 10

## I

icontrol (module), 12

icontrol.authtoken (module), 7

icontrol.exceptions (module), 8

icontrol.session (module), 8

iControlRESTSession (class in icontrol.session), 10

iControlRESTTokenAuth (class in icontrol.authtoken), 7

iControlUnexpectedHTTPError, 8

InvalidBigIP\_ICRURI, 8

InvalidInstanceNameOrFolder, 8

InvalidPrefixCollection, 8

InvalidScheme, 8

InvalidSuffixCollection, 8

InvalidURIComponentPart, 8

## P

patch() (icontrol.session.iControlRESTSession method), 11

post() (icontrol.session.iControlRESTSession method), 11

put() (icontrol.session.iControlRESTSession method), 11

## T

token (icontrol.session.iControlRESTSession attribute), 12