
Requests-JWT Documentation

Release 0.3

Thomas Grenfell Smith

Dec 11, 2018

Contents

Python Module Index

3

class `requests_jwt.JWTAuth` (*secret*, *alg=u'HS256'*, *header_format=u'JWT token=%s'*)
 An Authorization/Authentication system for requests, implementing JSON Web Tokens.

The basic usage is this:

```
auth = JWTAuth(secret)
# Maybe add more fields to the payload (see below)
resp = requests.get('http://example.com/', auth=auth)
```

You can add fields to the signed payload using the `expire()` and `add_field()` methods.

This is a 'Custom Authentication' mechanism for Kenneth Reitz's Requests library; see the [example in the docs](#) for some context.

For more on JSON Web Tokens, see [the standard](#).

See the documentation of PyJWT for the list of available algorithms.

add_field (*name*, *generator*)

Add a field to the JWT payload.

- *name*: The name of the field. Should be a string.
- *generator*: a value or generator, the value of the field.

If *generator* is callable, then each time a request is made with this JWTAuth, the generator will be called with one argument: a `PreparedRequest` object. See [this page](#) for a list of the available properties:

For instance, here is field that will have your JWT sign the path that it is requesting:

```
auth.add_field('path', lambda req: req.path_url)
```

If *generator* is not callable, it will be included directly in the JWT payload. It could be a string or a JSON-serializable object.

This module provides several payload fields ready to go: `payload_method()`, `payload_path()`, and `payload_body()`. `expire()` is also a wrapper around `add_field`.

expire (*secs*)

Adds the standard 'exp' field, used to prevent replay attacks.

Adds the 'exp' field to the payload. When a request is made, the field says that it should expire at now + *secs* seconds.

Of course, this provides no protection unless the server reads and interprets this field.

set_header_format (*new_format*)

Modify the contents of the `Authorization: header`. This must be a format string with one `%s` in it.

`requests_jwt.payload_method` (*req*)

A generator that will include the request method in the JWT payload.

```
>>> auth = JWTAuth('secret')
>>> auth.add_field('method', payload_method)
```

`requests_jwt.payload_path` (*req*)

A generator that will include the request's path ('/blah/index.html') in the JWT payload.

```
>>> auth = JWTAuth('secret')
>>> auth.add_field('path', payload_path)
```

`requests_jwt.payload_body` (*req*)

A generator that will include the sha256 signature of the request's body in the JWT payload. This is only done if the request could have a body: if the method is POST or PUT.

```
>>> auth = JWTAuth('secret')
>>> auth.add_field('body', payload_body)
```

r

requests_jwt, ??

A

add_field() (requests_jwt.JWTAuth method), 1

E

expire() (requests_jwt.JWTAuth method), 1

J

JWTAuth (class in requests_jwt), 1

P

payload_body() (in module requests_jwt), 1

payload_method() (in module requests_jwt), 1

payload_path() (in module requests_jwt), 1

R

requests_jwt (module), 1

S

set_header_format() (requests_jwt.JWTAuth method), 1