ursine Documentation

Release 0.3.1

Terry Kerr

May 23, 2018

Contents:

ursine is a SIP URI library intended to be easy to integrate into and use for other SIP-focused libraries such as aiosip. In particular its meant to be easy to do the following

- parse SIP URIs into easy to work with objects
- generate new SIP URIs
- normalize the representation of SIP URIs
- allowing comparisons / hashing of URIs

and do all the same as well for SIP header values with contained URIs.

CHAPTER 1

userinfo and hostport

The SIP spec often refers to the 'userinfo' and 'hostport' meaning the username+password and hostname+port respectively in a URI. For convenience ursine offers handling the userinfo and hostport either as a single entity or individually. The code below shows this with two effectively identical ways of creating the same URI.

```
from ursine import URI

uri1 = URI.build(scheme='sip',
                 host='localhost',
                 port=5080,
                 userinfo='john:pass')
uri2 = URI.build(scheme='sip',
                 hostport='localhost:5080',
                 user='john',
                 password='pass')
assert uri1 == uri2
```

1.1 URI

The URI object can be created from a parseable URI in a string, and unspecified attributes will take on reasonable defaults or, where applicable, defaults as specified in the SIP standard.

URIs are treated as immutable, though new URIs can be created from existing ones easily.

```
from ursine import URI

uri1 = URI('sip:localhost:5080;transport=tcp')
uri2 = URI('sips:localhost:5080')

assert uri1 != uri2
assert uri1 == uri2.with_scheme('sip')
# since URIs are immutable, uri2 is unchanged
assert uri1 != uri2
```

class ursine.uri.**URI**(*uri: str*)
A SIP URI

classmethod build(*, *scheme: str*, *user: Union[str, NoneType] = None*, *password: Union[str, NoneType] = None*, *userinfo: Union[str, NoneType] = None*, *host: Union[str, NoneType] = None*, *port: Union[int, NoneType] = None*, *hostport: Union[str, NoneType] = None*, *parameters: Union[dict, NoneType] = None*, *headers: Union[multidict._multidict.MultiDict, NoneType] = None*, *transport: Union[str, NoneType] = None*) → ursine.uri.URI
Build a URI from individual pieces.

Both the userinfo and hostport may be broken down into user/password and host/port respectively for convenience, and similarly the transport parameter is offered as an argument for convenience.

short_str()
Get a string representation without parameters/headers.

with_headers(*headers: multidict._multidict.MultiDict*)
Create a new URI from *self* with specific headers.

with_host(*host: str*)
Create a new URI from *self* with a specific host.

with_hostport(*hostport: str*)
Create a new URI from *self* with a specific hostport.

with_parameters(*parameters: Dict[str, str]*)
Create a new URI from *self* with specific parameters.

with_password(*password: Union[str, NoneType]*)
Create a new URI from *self* with a specific password.

with_port(*port: int*)
Create a new URI from *self* with a specific port.

with_scheme(*scheme: str*)
Create a new URI from *self* with a specific scheme.

with_transport(*transport: str*)
Create a new URI from *self* with a specific transport.

with_user(*user: Union[str, NoneType]*)
Create a new URI from *self* with a specific user.

with_userinfo(*userinfo: str*)
Create a new URI from *self* with a specific userinfo.

exception ursine.uri.**URIError**

1.2 Header

A SIP header wraps together a display name, sip URI, and header parameters, and so too does the ursine header, but the Header class also offers one utility - it can be used to ensure that a given header has a tag (randomly generating one and applying it if no tag exists and no tag to use is specified).

```
from ursine import Header, URI

hdr1 = Header('"Alice" <sip:localhost>;tag=foo')
hdr2 = Header('<sip:localhost>')
```

(continues on next page)

(continued from previous page)

```
assert hdr1 == hdr2.with_display_name('Alice').with_tag('foo')
assert hdr1.with_display_name(None) == hdr2.with_tag('foo')

uri = URI.build(scheme='sip', host='localhost')
header_with_tag = Header.build(uri=uri).with_tag()

assert header_with_tag.tag is not None
```

class ursine.header.**Header**(*header: str*)

A SIP Header (Contact/To/From).

classmethod build(**, uri: ursine.uri.URI, display_name: Union[str, NoneType] = None, parameters: Union[typing.Dict[str, str], NoneType] = None, tag: Union[str, NoneType] = None*) → ursine.header.Header

Build a new Header from kwargs.

with_display_name(*display_name: str*)

Create a new Header from *self* with a specific display name.

with_parameters(*parameters: Dict[str, str]*)

Create a new Header from *self* with specific parameters.

with_tag(*tag: Union[str, NoneType] = None*)

Create a new Header from *self* guaranteed to have a tag.

If tag is defined the resulting Header will always have the given tag value, but if tag is specied or defaulted to None a new Header with a randomly generated tag will be returned.

with_uri(*uri: ursine.uri.URI*)

Create a new Header from *self* with a specific URI.

exception ursine.header.**HeaderError**

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

u

Index