arcp Documentation

Release 0.1.0

Stian Soiland-Reyes

Feb 05, 2018

Contents:

CHAPTER 1

arcp

Create/parse arcp (Archive and Package) URIs.

This module provides functions for creating arcp URIs, which can be used for identifying or parsing hypermedia files packaged in an archive like a ZIP file:

```
>>> from arcp import *

>>> arcp_random()
'arcp://uuid,dcd6b1e8-b3a2-43c9-930b-0119cf0dc538/'

>>> arcp_random("/foaf.ttl", fragment="me")
'arcp://uuid,dcd6b1e8-b3a2-43c9-930b-0119cf0dc538/foaf.ttl#me'

>>> arcp_hash(b"Hello World!", "/folder/")
'arcp://ni,sha-256;f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk/folder/'

>>> arcp_location("http://example.com/data.zip", "/file.txt")
'arcp://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/file.txt'
```

arcp URLs can be used with `urllib.parse`, for instance using `urllib.parse.urljoin()` to resolve relative references:

```
>>> css = arcp.arcp_name("app.example.com", "css/style.css")
>>> urllib.parse.urljoin(css, "../fonts/foo.woff")
'arcp://name,app.example.com/fonts/foo.woff'
```

In addition this module provides functions that can be used to parse arcp URIs into its constituent fields:

```
>>> is_arcp_uri("arcp://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/file.txt")
True

>>> is_arcp_uri("http://example.com/t")
False

>>> u = parse_arcp("arcp://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/file.txt")
```

```
ARCPSplitResult(scheme='arcp',prefix='uuid',name='b7749d0b-0e47-5fc4-999d-f154abe68065
↪',
  uuid='b7749d0b-0e47-5fc4-999d-f154abe68065',path='/file.txt',query='',fragment='')

>>> u.path
'/file.txt'
>>> u.prefix
'uuid'
>>> u.uuid
UUID('b7749d0b-0e47-5fc4-999d-f154abe68065')
>>> u.uuid.version
5

>>> parse_arcp("arcp://ni,sha-256;f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk/folder/
↪").hash
('sha-256', '7f83b1657ff1fc53b92dc18148a1d65dfc2d4b1fa3d677284addd200126d9069')
```

The object returned from *parse_arcp()* is similar to `urllib.parse.ParseResult`, but contains additional properties `prefix`, `uuid`, `ni`, `hash` and `name`, some of which will be `None` depending on the arcp prefix.

The function *arcp.parse.urlparse()* can be imported as an alternative to `urllib.parse.urlparse()`. If the scheme is `arcp` then the extra arcp fields like `prefix`, `uuid`, `hash` and `name` are available as from *parse_arcp()*, otherwise the output is the same as from `urllib.parse.urlparse()`:

```
>>> from arcp.parse import urlparse
>>> urlparse("arcp://ni,sha-256;f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk/folder/
↪soup;sads")
ARCPParseResult(scheme='arcp',prefix='ni',
   name='sha-256;f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk',
   ni='sha-256;f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk',
   hash=('sha-256', '7f83b1657ff1fc53b92dc18148a1d65dfc2d4b1fa3d677284addd200126d9069
↪',
   path='/folder/soup;sads',query='',fragment='')
>>> urlparse("http://example.com/help?q=a")
ParseResult(scheme='http', netloc='example.com', path='/help', params='',
  query='q=a', fragment='')
```

arcp.**is_arcp_uri**(*uri*)
: Return True if the uri string uses the arcp scheme, otherwise False.

arcp.**parse_arcp**(*uri*)
: Parse an arcp URI string into its constituent parts.

  The returned object is similar to `urllib.parse.urlparse()` in that it is a tuple of `(scheme,netloc,path,params,query,fragment)` with equally named properties, but it also adds properties for arcp fields:

  - prefix – arcp authority prefix, e.g. “uuid”, “ni” or “name”, or None if prefix is missing
  - name – arcp authority without prefix, e.g. “a4889890-a50a-4f14-b4e7-5fd83683a2b5” or “example.com”
  - uuid – a `uuid.UUID` object if prefix is “uuid”, otherwise None
  - ni – the arcp alg-val value according to RFC6920 if prefix is “ni”, otherwise None
  - hash – the hash method and hash as a hexstring if prefix is “ni”, otherwise None

arcp.**arcp_uuid**(*uuid*, *path='/'*, *query=None*, *fragment=None*)
: Generate an arcp URI for the given uuid.

  Parameters:

- uuid – a uuid string or UUID instance identifying the archive, e.g. `58ca7fa6-be2f-48e4-8b69-e63fb0d929fe`
- path – Optional path within archive.
- query – Optional query component.
- fragment – Optional fragment component.

`arcp.`**`arcp_random`**(*path='/'*, *query=None*, *fragment=None*, *uuid=None*)
Generate an arcp URI using a random uuid.

Parameters:

- path – Optional path within archive.
- query – Optional query component.
- fragment – Optional fragment component.
- uuid – optional UUID v4 string or UUID instance

`arcp.`**`arcp_location`**(*location*, *path='/'*, *query=None*, *fragment=None*, *namespace=UUID('6ba7b811-9dad-11d1-80b4-00c04fd430c8')*)
Generate an arcp URI for a given archive location.

Parameters:

- location: URL or location of archive, e.g. `http://example.com/data.zip`
- path – Optional path within archive.
- query – Optional query component.
- fragment – Optional fragment component.
- namespace – optional namespace UUID for non-URL location.

`arcp.`**`arcp_name`**(*name*, *path='/'*, *query=None*, *fragment=None*)
Generate an arcp URI for a given archive name.

Parameters:

- name – Absolute DNS or package name, e.g. `app.example.com`
- path – Optional path within archive.
- query – Optional query component.
- fragment – Optional fragment component.
- namespace – optional namespace UUID for non-URL location.

`arcp.`**`arcp_hash`**(*bytes=b''*, *path='/'*, *query=None*, *fragment=None*, *hash=None*)
Generate an arcp URI for a given archive hash checksum.

Parameters:

- bytes – Optional bytes of archive to checksum
- path – Optional path within archive.
- query – Optional query component.
- fragment – Optional fragment component.
- hash – Optional hash instance from `hashlib.sha256()`

Either `bytes` or `hash` must be provided. The `hash` parameter can be provided to avoid representing the whole archive bytes in memory.

CHAPTER 2

arcp.generate

Generate arcp URIs with various prefixes.

As detailed in draft-soilandreyes-arcp, the choice of arcp _prefix_ depends on the uniqueness constraints required to identify the archive.

arcp_random() can be used for a fresh arcp URI based on a pseudo-random generator. Use `urllib.parse.urljoin()` to resolve paths within the same archive.

arcp_uuid() can be used with a pre-made UUID instance, for instance loaded from an archive's manifest or generated with `uuid.uuid4()`

arcp_location() can be used to identify an archive based on its location URL, facilitating a UUID v5 authority.

arcp_name() can be used to identify an archive based on its absolute DNS name or package name within an installation.

`arcp.generate.`**`arcp_hash`**(*bytes=b''*, *path='/'*, *query=None*, *fragment=None*, *hash=None*)

Generate an arcp URI for a given archive hash checksum.

Parameters:

- bytes – Optional bytes of archive to checksum
- path – Optional path within archive.
- query – Optional query component.
- fragment – Optional fragment component.
- hash – Optional hash instance from `hashlib.sha256()`

Either `bytes` or `hash` must be provided. The `hash` parameter can be provided to avoid representing the whole archive bytes in memory.

`arcp.generate.`**`arcp_location`**(*location*, *path='/'*, *query=None*, *fragment=None*, *namespace=UUID('6ba7b811-9dad-11d1-80b4-00c04fd430c8')*)

Generate an arcp URI for a given archive location.

Parameters:

- location: URL or location of archive, e.g. `http://example.com/data.zip`
- path – Optional path within archive.
- query – Optional query component.
- fragment – Optional fragment component.
- namespace – optional namespace UUID for non-URL location.

arcp.generate.**arcp_name**(*name*, *path='/'*, *query=None*, *fragment=None*)
Generate an arcp URI for a given archive name.

Parameters:

- name – Absolute DNS or package name, e.g. `app.example.com`
- path – Optional path within archive.
- query – Optional query component.
- fragment – Optional fragment component.
- namespace – optional namespace UUID for non-URL location.

arcp.generate.**arcp_random**(*path='/'*, *query=None*, *fragment=None*, *uuid=None*)
Generate an arcp URI using a random uuid.

Parameters:

- path – Optional path within archive.
- query – Optional query component.
- fragment – Optional fragment component.
- uuid – optional UUID v4 string or UUID instance

arcp.generate.**arcp_uuid**(*uuid*, *path='/'*, *query=None*, *fragment=None*)
Generate an arcp URI for the given uuid.

Parameters:

- uuid – a uuid string or UUID instance identifying the archive, e.g. `58ca7fa6-be2f-48e4-8b69-e63fb0d929fe`
- path – Optional path within archive.
- query – Optional query component.
- fragment – Optional fragment component.

CHAPTER 3

arcp.parse

Parse arcp URIs.

Use is_arcp_uri() to detect of an URI string is using the arcp: URI scheme, in which case parse_arcp() can be used to split it into its components.

The urlparse() function can be used as a replacement for urllib.parse.urlparse() - supporting any URIs. If the URI is using the arcp: URI scheme, additional components are available as from parse_arcp().

class arcp.parse.**ARCPParseResult**(**args*)

Result of parsing an arcp URI.

This class does not detect if the arcp URI was valid according to the specification.

This class extends `urllllib.parse.ParseResult` adding arcp properties, some of which may be *None*.

hash

A tuple (hash_method,hash_hex) if the prefix is "ni", otherwise None.

name

The URI's authority without arcp prefix.

ni

The arcp ni string if the prefix is "ni", otherwise None.

ni_uri(*authority=''*)

The ni URI (RFC6920) if the prefix is "ni", otherwise None.

If the `authority` parameter is provided, it will be used in the returned URI.

ni_well_known(*base=''*)

The ni .well-known URI (RFC5785) if the prefix is "ni", otherwise None.

The parameter `base`, if provided, should be an absolute URI like `"http://example.com/"` - a relative URI is returned otherwise.

nih_uri()

The nih URI (RFC6920) if the prefix is "ni", otherwise None.

prefix
The arcp prefix, e.g. “uuid”, “ni”, “name” or None if no prefix was present.

uuid
The arcp UUID if the prefix is “uuid”, otherwise None.

arcp.parse.**is_arcp_uri**(*uri*)
Return True if the uri string uses the arcp scheme, otherwise False.

arcp.parse.**parse_arcp**(*uri*)
Parse an arcp URI string into its constituent parts.

The returned object is similar to `urllib.parse.urlparse()` in that it is a tuple of `(scheme,netloc,path,params,query,fragment)` with equally named properties, but it also adds properties for arcp fields:

- prefix – arcp authority prefix, e.g. “uuid”, “ni” or “name”, or None if prefix is missing
- name – arcp authority without prefix, e.g. “a4889890-a50a-4f14-b4e7-5fd83683a2b5” or “example.com”
- uuid – a `uuid.UUID` object if prefix is “uuid”, otherwise None
- ni – the arcp alg-val value according to RFC6920 if prefix is “ni”, otherwise None
- hash – the hash method and hash as a hexstring if prefix is “ni”, otherwise None

arcp.parse.**urlparse**(*uri*)
Parse any URI string into constituent parts.

The returned object is similar to `urllib.parse.urlparse()` in that it is a tuple of `(scheme,netloc,path,params,query,fragment)` with equally named properties, but if the URI scheme is “arcp” this also adds arcp properties as in *parse_arcp()*.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

`arcp` provides functions for creating arcp URIs, which can be used for identifying or parsing hypermedia files packaged in an archive or package, like a ZIP file.

arcp URIs can be used to consume or reference hypermedia resources bundled inside a file archive or an application package, as well as to resolve URIs for archive resources within a programmatic framework.

This URI scheme provides mechanisms to generate a unique base URI to represent the root of the archive, so that relative URI references in a bundled resource can be resolved within the archive without having to extract the archive content on the local file system.

An arcp URI can be used for purposes of isolation (e.g. when consuming multiple archives), security constraints (avoiding "climb out" from the archive), or for externally identiyfing sub-resources referenced by hypermedia formats.

Examples:

- `arcp://uuid,32a423d6-52ab-47e3-a9cd-54f418a48571/doc.html`
- `arcp://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/pics/`
- `arcp://ni,sha-256;F-34D4TUeOfG0selz7REKRDo4XePkewPeQYtjL3vQs0/`
- `arcp://name,gallery.example.org/`

The different forms of URI authority in arcp URIs can be used depending on which uniqueness constraints to apply when addressing an archive. See the arcp specification (draft-soilandreyes-arcp) for details.

Note that this library only provides mechanisms to *generate* and *parse* arcp URIs, and do *not* integrate with any particular archive or URL handling modules like `zipfile` or `urllib.request`.

4.1 License

4.2 Source code and contributing

Source code: <https://github.com/stain/arcp-py>

Feel free to raise a pull request at <https://github.com/stain/arcp-py/pulls> or an issue at <https://github.com/stain/arcp-py/issues>.

4.3 Installing

You will need Python 2.7, Python 3.4 or later (Recommended: 3.6).

If you have pip, then the easiest is normally to install from <https://pypi.org/project/arcp/> using:

```
pip install arcp
```

If you want to install manually from this code base, then try:

```
python setup.py install
```

Python Module Index

a

Index