

---

# **uriutils Documentation**

*Release 0.1.13*

**Yanchuan Sim**

**Nov 02, 2018**



---

# Contents

---

<b>1</b>	<b>API Documentation</b>	<b>3</b>
1.1	Read / Write functions . . . . .	3
1.2	URI information . . . . .	4
1.3	Argument Parser types . . . . .	5
<b>2</b>	<b>Storages Documentation</b>	<b>7</b>
2.1	Local filesystem . . . . .	8
2.2	AWS Simple Storage Service . . . . .	10
2.3	Google Cloud Storage . . . . .	11
2.4	HTTP . . . . .	12
2.5	AWS Simple Notification Service . . . . .	13
<b>3</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>



Welcome to the documentation for *uriutils*. This package aims to make it transparent to the user and the developer the underlying storage system (i.e., S3, Google Cloud, local filesystems, etc) by wrapping the different protocols in a common interface.

Currently, the following storage systems are supported:

- Local filesystem (i.e., empty or `file` scheme)
- Amazon Web Services Simple Storage Services (S3) using `S3.Client` (i.e., `s3` scheme)
- Amazon Web Services Simple Notification Service (SNS) using `SNS.Client` (i.e., `sns` scheme)
- Google Cloud Storage using `google.cloud.storage.client` (i.e., `gcs` or `gs` scheme)
- HTTP using `requests` (i.e., `http` or `https` scheme)



## 1.1 Read / Write functions

`uriutils.uriutils.uri_open(uri, mode='rb', auto_compress=True, in_memory=True, delete_tempfile=True, textio_args={}, storage_args={})`

Opens a URI for reading / writing. Analogous to the `open()` function. This method supports `with` context handling:

```
with uri_open('http://www.example.com', mode='r') as f:
    print(f.read())
```

### Parameters

- **uri** (*str*) – URI of file to open
- **mode** (*str*) – Either `rb`, `r`, `w`, or `wb` for read/write modes in binary/text respectively
- **auto\_compress** (*bool*) – Whether to automatically use the `gzip` module with `.gz` URIsF
- **in\_memory** (*bool*) – Whether to store entire file in memory or in a local temporary file
- **delete\_tempfile** (*bool*) – When `in_memory` is `False`, whether to delete the temporary file on close
- **textio\_args** (*dict*) – Keyword arguments to pass to `io.TextIOWrapper` for text read/write mode
- **storage\_args** (*dict*) – Keyword arguments to pass to the underlying storage object

**Returns** file-like object to URI

`uriutils.uriutils.uri_read(*args, **kwargs)`

Reads the contents of a URI into a string or bytestring. See `uri_open()` for complete description of keyword parameters.

**Returns** Contents of URI

**Return type** `str, bytes`

`uriutils.uriutils.uri_dump(uri, content, mode='wb', **kwargs)`

Dumps the contents of a string/bytestring into a URI. See `uri_open()` for complete description of keyword parameters.

**Parameters**

- **uri** (`str`) – URI to dump contents to
- **content** (`str`) – Contents to write to URI
- **mode** (`str`) – Either `w`, or `wb` to write binary/text content respectively

## 1.2 URI information

`uriutils.uriutils.uri_exists(uri, storage_args={})`

Check if URI exists.

**Parameters**

- **uri** (`str`) – URI to check existence
- **storage\_args** (`dict`) – Keyword arguments to pass to the underlying storage object

**Returns** `True` if URI exists

**Return type** `bool`

`uriutils.uriutils.uri_exists_wait(uri, timeout=300, interval=5, storage_args={})`

Block / waits until URI exists.

**Parameters**

- **uri** (`str`) – URI to check existence
- **timeout** (`float`) – Number of seconds before timing out
- **interval** (`float`) – Calls `uri_exists()` every interval seconds
- **storage\_args** (`dict`) – Keyword arguments to pass to the underlying storage object

**Returns** `True` if URI exists

**Return type** `bool`

`uriutils.uriutils.get_uri_metadata(uri, storage_args={})`

Get the “metadata” from URI. This is most commonly used with bucket storage on the Cloud such as S3 and Google Cloud.

**Parameters**

- **uri** (`str`) – URI to get metadata for
- **storage\_args** (`dict`) – Keyword arguments to pass to the underlying storage object

**Returns** Metadata associated with URI

**Return type** `dict`

`uriutils.uriutils.get_uri_obj(uri, storage_args={})`

Retrieve the underlying storage object based on the URI (i.e., scheme).

**Parameters**

- **uri** (`str`) – URI to get storage object for



- **storage\_args** (*dict*) – Keyword arguments to pass to the underlying storage object

## 1.3 Argument Parser types

**class** `uriutils.uriutils.URIType`

A convenience class that can be used as the `type` argument to `argparse.ArgumentParser.add_argument()`. It will return the result of `urllib.parse.urlparse()`.

**class** `uriutils.uriutils.URIFileType` (*mode='rb', \*\*kwargs*)

A convenience class that can be used as the `type` argument to `argparse.ArgumentParser.add_argument()`. It will return a file-like object using `uri_open()`.

See `uri_open()` for complete description of keyword parameters.

**class** `uriutils.uriutils.URIDirType` (*create=False, storage\_args={}*)

A convenience class that can be used as the `type` argument to `argparse.ArgumentParser.add_argument()`. It will return the result of `urllib.parse.urlparse()`.

### Parameters

- **create** (*bool*) – Whether to create directory (and thus “ensure” that directory exists)
- **storage\_args** (*dict*) – Keyword arguments to pass to the underlying storage object



---

## Storages Documentation

---

This module defines all the storage systems supported by uriutils.

**class** `uriutils.storages.URIBytesOutput` (*uri\_obj*)

A BytesIO object for output that flushes content to the remote URI on close.

**\_\_init\_\_** (*uri\_obj*)

*x.\_\_init\_\_(...)* initializes *x*; see `help(type(x))` for signature

**close** () → None. Disable all I/O operations.

**name**

**class** `uriutils.storages.BaseURI` (*storage\_args={}*)

This is the base URI storage object that is inherited by the different storage systems. It defines the methods and operations that can be “conducted” on a URI. Almost all of these methods have to be implemented by a storage class.

**SUPPORTED\_SCHEMES** = []

Defines the schemes supported by this storage system.

**VALID\_STORAGE\_ARGS** = []

The set of *storage\_args* keyword arguments that is handled by this storage system.

**\_\_init\_\_** (*storage\_args={}*)

**Parameters** *storage\_args* (*dict*) – Arguments that will be applied to the storage system for read/write operations

**dir\_exists** ()

Check if the URI exists as a directory.

**Returns** `True` if URI exists as a directory

**Return type** `bool`

**download\_file** (*filename*)

Download the binary content stored in the URI for this object directly to local file.

**Parameters** *filename* (*str*) – Filename on local filesystem

**exists** ()

**Returns** True if URI exists

**Return type** bool

**get\_content** ()

**Returns** the bytestring stored at this object's URI

**Return type** bytes

**get\_metadata** ()

**Returns** the metadata associated with this object's URI

**Return type** dict

**join** (*path*)

Similar to `os.path.join()` but returns a storage object instead.

**Parameters** **path** (*str*) – path to join on to this object's URI

**Returns** a storage object

**Return type** *BaseURI*

**list\_dir** ()

List the contents of a directory.

**make\_dir** ()

Create a directory.

**classmethod parse\_uri** (*uri*, *storage\_args*={})

Parses the URI and return an instantiation of the storage system if it is supported.

**Parameters**

- **uri** (*str*) – URI to check
- **storage\_args** (*dict*) – Keyword arguments to pass to the underlying storage object

**Returns** None if this storage system does not support *uri*.

**put\_content** (*content*)

**Parameters** **content** (*bytes*) – Content to write to this object's URI

**upload\_file** (*filename*)

Upload the binary content in *filename* to the URI for this object.

**Parameters** **filename** (*str*) – Filename on local filesystem

## 2.1 Local filesystem

**class** `uriutils.storages.FileURI` (*filepath*, *storage\_args*={})

Storage system for local filesystem.

**Parameters**

- **filepath** (*str*) – Local file path
- **storage\_args** (*dict*) – Keyword arguments that are passed to `open()`

**SUPPORTED\_SCHEMES** = `set(['', 'file'])`

Supported schemes for *FileURI*.

**VALID\_STORAGE\_ARGS** = ['mode', 'buffering', 'encoding', 'errors', 'newline', 'closefd',  
Storage arguments allowed to pass to open () methods.

**\_\_init\_\_** (*filepath*, *storage\_args*={})

**Parameters** **storage\_args** (*dict*) – Arguments that will be applied to the storage system for read/write operations

**\_\_str\_\_** ()

**Returns** a nicely formed URI for this object.

**dir\_exists** ()

Check if the URI exists as a directory.

**Returns** `True` if URI exists as a directory

**Return type** `bool`

**download\_file** (*filename*)

Download the binary content stored in the URI for this object directly to local file.

**Parameters** **filename** (*str*) – Filename on local filesystem

**exists** ()

**Returns** `True` if URI exists

**Return type** `bool`

**get\_content** ()

**Returns** the bytestring stored at this object's URI

**Return type** `bytes`

**list\_dir** ()

List the contents of a directory.

**make\_dir** ()

Create a directory.

**classmethod parse\_uri** (*uri*, *storage\_args*={})

Parses the URI and return an instantiation of the storage system if it is supported.

**Parameters**

- **uri** (*str*) – URI to check
- **storage\_args** (*dict*) – Keyword arguments to pass to the underlying storage object

**Returns** `None` if this storage system does not support *uri*.

**put\_content** (*content*)

**Parameters** **content** (*bytes*) – Content to write to this object's URI

**upload\_file** (*filename*)

Upload the binary content in *filename* to the URI for this object.

**Parameters** **filename** (*str*) – Filename on local filesystem

## 2.2 AWS Simple Storage Service

**class** `uriutils.storages.S3URI` (*bucket, key, storage\_args={}*)

Storage system for AWS S3.

**SUPPORTED\_SCHEMES** = `set(['s3'])`

Supported schemes for *S3URI*.

**VALID\_STORAGE\_ARGS** = `['ACL', 'CacheControl', 'ContentDisposition', 'ContentEncoding',`

`Storage arguments allowed to pass to S3.Client methods.`

**\_\_init\_\_** (*bucket, key, storage\_args={}*)

### Parameters

- **bucket** (*str*) – Bucket name
- **key** (*str*) – Key to file
- **storage\_args** (*dict*) – Keyword arguments that are passed to *S3.Client*

**\_\_str\_\_** ()

**Returns** a nicely formed URI for this object.

**dir\_exists** ()

Check if the URI exists as a directory.

**Returns** `True` if URI exists as a directory

**Return type** `bool`

**download\_file** (*filename*)

Download the binary content stored in the URI for this object directly to local file.

**Parameters** **filename** (*str*) – Filename on local filesystem

**exists** ()

Uses HEAD requests for efficiency.

**get\_content** ()

**Returns** the bytestring stored at this object's URI

**Return type** `bytes`

**get\_metadata** ()

Uses HEAD requests for efficiency.

**list\_dir** ()

Non-recursive file listing.

**Returns** A generator over files in this “directory” for efficiency.

**make\_dir** ()

Ignored for S3.

**classmethod** **parse\_uri** (*uri, storage\_args={}*)

Parses the URI and return an instantiation of the storage system if it is supported.

### Parameters

- **uri** (*str*) – URI to check
- **storage\_args** (*dict*) – Keyword arguments to pass to the underlying storage object

**Returns** `None` if this storage system does not support *uri*.

`put_content` (*content*)

**Parameters** `content` (*bytes*) – Content to write to this object’s URI

`upload_file` (*filename*)

Upload the binary content in `filename` to the URI for this object.

**Parameters** `filename` (*str*) – Filename on local filesystem

## 2.3 Google Cloud Storage

**class** `uriutils.storages.GoogleCloudStorageURI` (*bucket, key, storage\_args={}*)

Storage system for Google Cloud storage.

**SUPPORTED\_SCHEMES** = `set(['gcs', 'gs'])`

Supported schemes for `GoogleCloudStorageURI`.

**VALID\_STORAGE\_ARGS** = `['chunk_size', 'encryption_key']`

Storage arguments allowed to pass to `google.cloud.storage.client` methods.

`__init__` (*bucket, key, storage\_args={}*)

**Parameters**

- **bucket** (*str*) – Bucket name
- **key** (*str*) – Key to file
- **storage\_args** (*dict*) – Keyword arguments that are passed to `google.cloud.storage.client`

`__str__` ()

**Returns** a nicely formed URI for this object.

`dir_exists` ()

Check if the URI exists as a directory.

**Returns** `True` if URI exists as a directory

**Return type** `bool`

`download_file` (*filename*)

Download the binary content stored in the URI for this object directly to local file.

**Parameters** `filename` (*str*) – Filename on local filesystem

`exists` ()

Uses HEAD requests for efficiency.

`get_content` ()

**Returns** the bytestring stored at this object’s URI

**Return type** `bytes`

`get_metadata` ()

Uses HEAD requests for efficiency.

`list_dir` ()

Non-recursive file listing.

**Returns** A generator over files in this “directory” for efficiency.

**make\_dir()**

Create a directory.

**classmethod parse\_uri** (*uri*, *storage\_args*={})

Parses the URI and return an instantiation of the storage system if it is supported.

**Parameters**

- **uri** (*str*) – URI to check
- **storage\_args** (*dict*) – Keyword arguments to pass to the underlying storage object

**Returns** None if this storage system does not support *uri*.

**put\_content** (*content*)

The default content type is set to `application/octet-stream` and content encoding set to `None`.

**upload\_file** (*filename*)

Upload the binary content in *filename* to the URI for this object.

**Parameters** **filename** (*str*) – Filename on local filesystem

## 2.4 HTTP

**class** `uriutils.storages.HTTPURI` (*url*, *raise\_for\_status*=*True*, *method*=*None*, *storage\_args*={})

Storage system for HTTP/HTTPS.

**SUPPORTED\_SCHEMES** = `set(['http', 'https'])`

Supported schemes for `HTTPURI`.

**VALID\_STORAGE\_ARGS** = `['params', 'headers', 'cookies', 'auth', 'timeout', 'allow_redire`

Keyword arguments passed to `requests.request()`.

**\_\_init\_\_** (*url*, *raise\_for\_status*=*True*, *method*=*None*, *storage\_args*={})

**Parameters**

- **uri** (*str*) – HTTP URI.
- **raise\_for\_status** (*str*) – Raises a `requests.RequestException` when the response status code is not 2xx (i.e., calls `requests.Request.raise_for_status()`)
- **method** (*str*) – Overrides the default method for all HTTP operations.
- **storage\_args** (*dict*) – Keyword arguments that are passed to `requests.request()`

**\_\_str\_\_** ()

**Returns** a nicely formed URI for this object.

**dir\_exists** ()

Makes a HEAD requests to the URI.

**Returns** `True` if status code is 2xx.

**download\_file** (*filename*)

Download the binary content stored in the URI for this object directly to local file.

**Parameters** **filename** (*str*) – Filename on local filesystem

**exists** ()



**Returns** True if URI exists

**Return type** bool

**get\_content** ()

**Returns** the bytestring stored at this object's URI

**Return type** bytes

**make\_dir** ()

Ignored.

**classmethod parse\_uri** (*uri*, *storage\_args*={})

Parses the URI and return an instantiation of the storage system if it is supported.

**Parameters**

- **uri** (*str*) – URI to check
- **storage\_args** (*dict*) – Keyword arguments to pass to the underlying storage object

**Returns** None if this storage system does not support *uri*.

**put\_content** (*content*)

Makes a PUT request with the content in the body.

**Raise** An `requests.RequestException` if it is not 2xx.

**upload\_file** (*filename*)

Upload the binary content in *filename* to the URI for this object.

**Parameters** **filename** (*str*) – Filename on local filesystem

## 2.5 AWS Simple Notification Service

**class** `uriutils.storages.SNSURI` (*topic\_name*, *region*, *storage\_args*={})

Storage system for AWS Simple Notification Service.

**SUPPORTED\_SCHEMES** = set(['sns'])

Supported schemes for `SNSURI`.

**VALID\_STORAGE\_ARGS** = ['Subject', 'MessageAttributes', 'MessageStructure']

Keyword arguments passed to `SNS.Client.publish()`.

**\_\_init\_\_** (*topic\_name*, *region*, *storage\_args*={})

**Parameters**

- **topic\_name** (*str*) – Name of SNS topic for publishing; it can be either an ARN or just the topic name (thus defaulting to the current role's account)
- **region** (*str*) – AWS region of SNS topic (defaults to current role's region)
- **storage\_args** (*dict*) – Keyword arguments that are passed to `SNS.Client.publish()`

**\_\_str\_\_** ()

**Returns** a nicely formed URI for this object.

**dir\_exists** ()

Not supported.

**download\_file** (*filename*)

Not supported.

**exists** ()

**Returns** True if the SNS topic exists

**get\_content** ()

Not supported.

**make\_dir** ()

Create a directory.

**classmethod parse\_uri** (*uri*, *storage\_args*={})

Parses the URI and return an instantiation of the storage system if it is supported.

**Parameters**

- **uri** (*str*) – URI to check
- **storage\_args** (*dict*) – Keyword arguments to pass to the underlying storage object

**Returns** None if this storage system does not support *uri*.

**put\_content** (*content*)

Publishes a message straight to SNS.

**Parameters content** (*bytes*) – raw bytes content to publish, will decode to UTF-8 if string is detected

**upload\_file** (*filename*)

Upload the binary content in *filename* to the URI for this object.

**Parameters filename** (*str*) – Filename on local filesystem

## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `search`



**u**

`uriutils.storages`, 7



## Symbols

[\\_\\_init\\_\\_\(\)](#) (uriutils.storages.BaseURI method), 7  
[\\_\\_init\\_\\_\(\)](#) (uriutils.storages.FileURI method), 9  
[\\_\\_init\\_\\_\(\)](#) (uriutils.storages.GoogleCloudStorageURI method), 11  
[\\_\\_init\\_\\_\(\)](#) (uriutils.storages.HTTPURI method), 12  
[\\_\\_init\\_\\_\(\)](#) (uriutils.storages.S3URI method), 10  
[\\_\\_init\\_\\_\(\)](#) (uriutils.storages.SNSURI method), 13  
[\\_\\_init\\_\\_\(\)](#) (uriutils.storages.URIBytesOutput method), 7  
[\\_\\_str\\_\\_\(\)](#) (uriutils.storages.FileURI method), 9  
[\\_\\_str\\_\\_\(\)](#) (uriutils.storages.GoogleCloudStorageURI method), 11  
[\\_\\_str\\_\\_\(\)](#) (uriutils.storages.HTTPURI method), 12  
[\\_\\_str\\_\\_\(\)](#) (uriutils.storages.S3URI method), 10  
[\\_\\_str\\_\\_\(\)](#) (uriutils.storages.SNSURI method), 13

## B

BaseURI (class in uriutils.storages), 7

## C

[close\(\)](#) (uriutils.storages.URIBytesOutput method), 7

## D

[dir\\_exists\(\)](#) (uriutils.storages.BaseURI method), 7  
[dir\\_exists\(\)](#) (uriutils.storages.FileURI method), 9  
[dir\\_exists\(\)](#) (uriutils.storages.GoogleCloudStorageURI method), 11  
[dir\\_exists\(\)](#) (uriutils.storages.HTTPURI method), 12  
[dir\\_exists\(\)](#) (uriutils.storages.S3URI method), 10  
[dir\\_exists\(\)](#) (uriutils.storages.SNSURI method), 13  
[download\\_file\(\)](#) (uriutils.storages.BaseURI method), 7  
[download\\_file\(\)](#) (uriutils.storages.FileURI method), 9  
[download\\_file\(\)](#) (uriutils.storages.GoogleCloudStorageURI method), 11  
[download\\_file\(\)](#) (uriutils.storages.HTTPURI method), 12  
[download\\_file\(\)](#) (uriutils.storages.S3URI method), 10  
[download\\_file\(\)](#) (uriutils.storages.SNSURI method), 13

## E

[exists\(\)](#) (uriutils.storages.BaseURI method), 7

[exists\(\)](#) (uriutils.storages.FileURI method), 9  
[exists\(\)](#) (uriutils.storages.GoogleCloudStorageURI method), 11  
[exists\(\)](#) (uriutils.storages.HTTPURI method), 12  
[exists\(\)](#) (uriutils.storages.S3URI method), 10  
[exists\(\)](#) (uriutils.storages.SNSURI method), 14

## F

FileURI (class in uriutils.storages), 8

## G

[get\\_content\(\)](#) (uriutils.storages.BaseURI method), 8  
[get\\_content\(\)](#) (uriutils.storages.FileURI method), 9  
[get\\_content\(\)](#) (uriutils.storages.GoogleCloudStorageURI method), 11  
[get\\_content\(\)](#) (uriutils.storages.HTTPURI method), 13  
[get\\_content\(\)](#) (uriutils.storages.S3URI method), 10  
[get\\_content\(\)](#) (uriutils.storages.SNSURI method), 14  
[get\\_metadata\(\)](#) (uriutils.storages.BaseURI method), 8  
[get\\_metadata\(\)](#) (uriutils.storages.GoogleCloudStorageURI method), 11  
[get\\_metadata\(\)](#) (uriutils.storages.S3URI method), 10  
[get\\_uri\\_metadata\(\)](#) (in module uriutils.uriutils), 4  
[get\\_uri\\_obj\(\)](#) (in module uriutils.uriutils), 4  
 GoogleCloudStorageURI (class in uriutils.storages), 11

## H

HTTPURI (class in uriutils.storages), 12

## J

[join\(\)](#) (uriutils.storages.BaseURI method), 8

## L

[list\\_dir\(\)](#) (uriutils.storages.BaseURI method), 8  
[list\\_dir\(\)](#) (uriutils.storages.FileURI method), 9  
[list\\_dir\(\)](#) (uriutils.storages.GoogleCloudStorageURI method), 11  
[list\\_dir\(\)](#) (uriutils.storages.S3URI method), 10

## M

make\_dir() (uriutils.storages.BaseURI method), 8  
make\_dir() (uriutils.storages.FileURI method), 9  
make\_dir() (uriutils.storages.GoogleCloudStorageURI method), 11  
make\_dir() (uriutils.storages.HTTPURI method), 13  
make\_dir() (uriutils.storages.S3URI method), 10  
make\_dir() (uriutils.storages.SNSURI method), 14

## N

name (uriutils.storages.URIBytesOutput attribute), 7

## P

parse\_uri() (uriutils.storages.BaseURI class method), 8  
parse\_uri() (uriutils.storages.FileURI class method), 9  
parse\_uri() (uriutils.storages.GoogleCloudStorageURI class method), 12  
parse\_uri() (uriutils.storages.HTTPURI class method), 13  
parse\_uri() (uriutils.storages.S3URI class method), 10  
parse\_uri() (uriutils.storages.SNSURI class method), 14  
put\_content() (uriutils.storages.BaseURI method), 8  
put\_content() (uriutils.storages.FileURI method), 9  
put\_content() (uriutils.storages.GoogleCloudStorageURI method), 12  
put\_content() (uriutils.storages.HTTPURI method), 13  
put\_content() (uriutils.storages.S3URI method), 10  
put\_content() (uriutils.storages.SNSURI method), 14

## S

S3URI (class in uriutils.storages), 10  
SNSURI (class in uriutils.storages), 13  
SUPPORTED\_SCHEMES (uriutils.storages.BaseURI attribute), 7  
SUPPORTED\_SCHEMES (uriutils.storages.FileURI attribute), 8  
SUPPORTED\_SCHEMES (uriutils.storages.GoogleCloudStorageURI attribute), 11  
SUPPORTED\_SCHEMES (uriutils.storages.HTTPURI attribute), 12  
SUPPORTED\_SCHEMES (uriutils.storages.S3URI attribute), 10  
SUPPORTED\_SCHEMES (uriutils.storages.SNSURI attribute), 13

## U

upload\_file() (uriutils.storages.BaseURI method), 8  
upload\_file() (uriutils.storages.FileURI method), 9  
upload\_file() (uriutils.storages.GoogleCloudStorageURI method), 12  
upload\_file() (uriutils.storages.HTTPURI method), 13  
upload\_file() (uriutils.storages.S3URI method), 11  
upload\_file() (uriutils.storages.SNSURI method), 14

uri\_dump() (in module uriutils.uriutils), 4  
uri\_exists() (in module uriutils.uriutils), 4  
uri\_exists\_wait() (in module uriutils.uriutils), 4  
uri\_open() (in module uriutils.uriutils), 3  
uri\_read() (in module uriutils.uriutils), 3  
URIBytesOutput (class in uriutils.storages), 7  
URIDirType (class in uriutils.uriutils), 5  
URIFileType (class in uriutils.uriutils), 5  
URIType (class in uriutils.uriutils), 5  
uriutils.storages (module), 7

## V

VALID\_STORAGE\_ARGS (uriutils.storages.BaseURI attribute), 7  
VALID\_STORAGE\_ARGS (uriutils.storages.FileURI attribute), 8  
VALID\_STORAGE\_ARGS (uriutils.storages.GoogleCloudStorageURI attribute), 11  
VALID\_STORAGE\_ARGS (uriutils.storages.HTTPURI attribute), 12  
VALID\_STORAGE\_ARGS (uriutils.storages.S3URI attribute), 10  
VALID\_STORAGE\_ARGS (uriutils.storages.SNSURI attribute), 13