uriutils Documentation

Release 0.1.13

Yanchuan Sim

Nov 02, 2018

Contents

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)

CHAPTER 1

API Documentation

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 respectiely
- **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 respectiely

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

CHAPTER 2

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

Python Module Index

Index