alluxio Documentation

Release 0.1.1

Alluxio, Inc

Feb 07, 2019

Contents

CHAPTER 1

Client Interface

Alluxio client, reader, and writer.

This module contains the Alluxio Client. The Reader and Writer are returned by certain I/O methods of the Client, they are not intended to be created by the API user.

The Client is based on Alluxio proxy server's REST API, all HTTP requests are handled by the requests library.

class alluxio.**Client**(*host*, *port*, *timeout=1800*)

Alluxio client.

Parameters

- **host** (*str*) – Alluxio proxy server's hostname.
- **port** (*int*) – Alluxio proxy server's web port.
- **timeout** (*int, optional*) – Seconds to wait for the REST server to respond before giving up. Defaults to 1800.

close(*file_id*)

Close a file.

When calling *open()* using a with statement, this method is automatically invoked when exiting the with block.

Parameters **file_id** (*int*) – The file ID returned by *open_file()* or *create_file()*.

Raises

- *alluxio.exceptions.NotFoundError* – If the path does not exist.
- *alluxio.exceptions.AlluxioError* – For any other exceptions thrown by Alluxio servers. Check the error status for additional details.
- *alluxio.exceptions.HTTPError* – If the underlying HTTP client library raises an error.

create_directory(*path*, *opt=None*)

Create a directory in Alluxio.

By default, the create directory operation enforces that the parent of the given path must exist and the path itself does not already exist. The directory will be created with access mode bits 'drwxr-xr-x'. The created directory will only exist in Alluxio and not in any of its under storages. You can change the behavior by setting optional parameters in kwargs.

Parameters

- **path** (*str*) – The path of the directory to be created.
- **opt** (*alluxio.option.CreateDirectory*) – Options to be used when creating a directory.

Raises

- *alluxio.exceptions.AlreadyExistsError* – If there is already a file or directory at the given path.
- *alluxio.exceptions.AlluxioError* – For any other exceptions thrown by Alluxio servers. Check the error status for additional details.
- *alluxio.exceptions.HTTPError* – If the underlying HTTP client library raises an error.

Examples

Create a directory recursively:

```
>>> opt = alluxio.option.CreateDirectory(recursive=True)
>>> create_directory('/parent/child/', opt)
```

Create a directory recursively and persist to under storage:

```
>>> opt = alluxio.option.CreateDirectory(recursive=True, write_type=wire.
↪WRITE_TYPE_CACHE_THROUGH)
>>> create_directory('/parent/child/', opt)
```

create_file(*path*, *opt=None*)

Create a file in Alluxio.

The file must not already exist and must be closed by calling *alluxio.Client.close()*.

A preferred way to write to a file is to use *open()*, see its documentation for details.

Parameters

- **path** (*str*) – The Alluxio path.
- **opt** (*alluxio.option.CreateFile*) – Options to be used when creating a file.

Returns The file ID, which can be passed to *alluxio.Client.write()* and *alluxio.Client.close()*.

Return type int

Raises

- *alluxio.exceptions.AlreadyExistsError* – If there is already a file or directory at the given path.
- *alluxio.exceptions.InvalidArgumentError* – If the path is invalid.
- *alluxio.exceptions.AlluxioError* – For any other exceptions thrown by Alluxio servers. Check the error status for additional details.

- *alluxio.exceptions.HTTPError* – If the underlying HTTP client library raises an error.

Examples

Create a file and write a string to it both in Alluxio and the under storage, finally close it:

```
>>> opt = alluxio.option.CreateFile(write_type=wire.WRITE_TYPE_CACHE_THROUGH)
>>> file_id = create_file('/file', opt)
>>> writer = write(file_id)
>>> writer.write('data')
>>> writer.close()
>>> close(file_id)
```

delete (*path*, *opt=None*)
Delete a directory or file in Alluxio.

By default, if path is a directory which contains files or directories, this method will fail. You can change the behavior by setting optional parameters in kwargs.

Parameters

- **path** (*str*) – The path of the directory or file to be deleted.
- **opt** (*alluxio.option.Delete*) – Options to be used when deleting a path.

Raises

- *alluxio.exceptions.NotFoundError* – If the path does not exist.
- *alluxio.exceptions.AlluxioError* – For any other exceptions thrown by Alluxio servers. Check the error status for additional details.
- *alluxio.exceptions.HTTPError* – If the underlying HTTP client library raises an error.

exists (*path*, *opt=None*)
Check whether a path exists in Alluxio.

Parameters

- **path** (*str*) – The Alluxio path.
- **opt** (*alluxio.option.Exists*) – Options to be used when checking whether a path exists.

Returns True if the path exists, False otherwise.

Return type bool

Raises

- *alluxio.exceptions.InvalidArgumentError* – If the path is invalid.
- *alluxio.exceptions.AlluxioError* – For any other exceptions thrown by Alluxio servers. Check the error status for additional details.
- *alluxio.exceptions.HTTPError* – If the underlying HTTP client library raises an error.

free (*path*, *opt=None*)
Free a file or directory from Alluxio.

By default, if the given path is a directory, its files and contained directories won't be freed recursively. You can change the behavior by setting optional parameters in kwargs.

Parameters

- **path** (*str*) – The Alluxio path.
- **opt** (*alluxio.option.Free*) – Options to be used when freeing a path.

Raises

- *alluxio.exceptions.NotFoundError* – If the path does not exist.
- *alluxio.exceptions.AlluxioError* – For any other exceptions thrown by Alluxio servers. Check the error status for additional details.
- *alluxio.exceptions.HTTPError* – If the underlying HTTP client library raises an error.

get_status(*path*, *opt=None*)

Get the status of a file or directory at the given path.

Parameters

- **path** (*str*) – The Alluxio path.
- **opt** (*alluxio.option.GetStatus*) – Options to be used when getting the status of a path.

Returns The information of the file or directory.

Return type *alluxio.wire.FileInfo*

Raises

- *alluxio.exceptions.NotFoundError* – If the path does not exist.
- *alluxio.exceptions.AlluxioError* – For any other exceptions thrown by Alluxio servers. Check the error status for additional details.
- *alluxio.exceptions.HTTPError* – If the underlying HTTP client library raises an error.

list_status(*path*, *opt=None*)

List the status of a file or directory at the given path.

Parameters

- **path** (*str*) – The Alluxio path, which should be a directory.
- **opt** (*alluxio.option.ListStatus*) – Options to be used when listing status.

Returns List of information of files and direcotries under path.

Return type List of *alluxio.wire.FileInfo*

Raises

- *alluxio.exceptions.NotFoundError* – If the path does not exist.
- *alluxio.exceptions.AlluxioError* – For any other exceptions thrown by Alluxio servers. Check the error status for additional details.
- *alluxio.exceptions.HTTPError* – If the underlying HTTP client library raises an error.

ls(*path*, *opt=None*)
List the names of the files and directories under path.

To get more information of the files and directories under path, call *list_status()*.

Parameters

- **path** (*str*) – The Alluxio path, which should be a directory.
- **opt** (*alluxio.option.ListStatus*) – Options to be used when listing status.

Returns A list of names of the files and directories under path.

Return type List of str

Raises

- *alluxio.exceptions.NotFoundError* – If the path does not exist.
- *alluxio.exceptions.AlluxioError* – For any other exceptions thrown by Alluxio servers. Check the error status for additional details.
- *alluxio.exceptions.HTTPError* – If the underlying HTTP client library raises an error.

mount(*path*, *src*, *opt=None*)
Mount an under storage specified by src to path in Alluxio.

Additional information or configuration, such as AWS credentials for mounting a S3 bucket or mounting the under storage in read only mode, can be provided by setting optional parameters in kwargs.

Parameters

- **path** (*str*) – The Alluxio path to be mounted to.
- **src** (*str*) – The under storage endpoint to mount.
- **opt** (*alluxio.option.Mount*) – Options to be used when mounting an under storage.

Raises

- *alluxio.exceptions.AlluxioError* – For any other exceptions thrown by Alluxio servers. Check the error status for additional details.
- *alluxio.exceptions.HTTPError* – If the underlying HTTP client library raises an error.

open(***kwds*)
Open a file for reading or writing.

It should be called using a with statement so that the reader or writer will be automatically closed.

Parameters

- **path** (*str*) – The Alluxio file to be read from or written to.
- **mode** (*str*) – Either 'r' for reading or 'w' for writing.
- **opt** – For reading, it is *alluxio.option.OpenFile*, for writing, it is *alluxio.option.CreateFile*.

Raises

- ValueError – If mode is neither 'w' nor 'r'.
- *alluxio.exceptions.InvalidArgumentError* – If the path is invalid.

- *alluxio.exceptions.NotFoundError* – If mode is 'r' but the path does not exist.
- *alluxio.exceptions.AlreadyExistsError* – If mode is 'w' but the path already exists.
- *alluxio.exceptions.AlluxioError* – For any other exceptions thrown by Alluxio servers. Check the error status for additional details.
- *alluxio.exceptions.HTTPError* – If the underlying HTTP client library raises an error.

Examples

Write a string to a file in Alluxio:

```
>>> with open('/file', 'w') as f:
>>>     f.write('data')
```

Copy a file in local filesystem to a file in Alluxio and also persist it into Alluxio's under storage, note that the second :func"*open* is python's built-in function:

```
>>> opt = alluxio.option.CreateFile(write_type=wire.WRITE_TYPE_CACHE_THROUGH)
>>> with alluxio_client.open('/alluxio-file', 'w', opt) as alluxio_file:
>>>     with open('/local-file', 'rb') as local_file:
>>>         alluxio_file.write(local_file)
```

Read the first 10 bytes of a file from Alluxio:

```
>>> with open('/file', 'r') as f:
>>>     print f.read(10)
```

open_file(*path*, *opt=None*)

Open a file in Alluxio for reading.

The file must be closed by calling *alluxio.Client.close()*.

The preferred way to read a file is to use *open()*.

Parameters

- **path** (*str*) – The Alluxio path.
- **opt** (*alluxio.option.OpenFile*) – Options to be used when opening a file.

Returns The file ID, which can be passed to *alluxio.Client.read()* and *alluxio.Client.close()*.

Return type int

Raises

- *alluxio.exceptions.NotFoundError* – If the path does not exist.
- *alluxio.exceptions.AlluxioError* – For any other exceptions thrown by Alluxio servers. Check the error status for additional details.
- *alluxio.exceptions.HTTPError* – If the underlying HTTP client library raises an error.

Examples

Open a file, read its contents, and close it:

```
>>> file_id = open_file('/file')
>>> reader = read(file_id)
>>> reader.read()
>>> reader.close()
>>> close(file_id)
```

read(*file_id*)
Creates a `Reader` for reading a file.

Parameters **file_id** (*int*) – The file ID returned by *open_file()*.

Returns The reader for reading the file as a stream.

Return type *Reader*

rename(*path*, *dst*, *opt=None*)
Rename path to dst in Alluxio.

Parameters

- **path** (*str*) – The Alluxio path to be renamed.
- **dst** (*str*) – The Alluxio path to be renamed to.
- **opt** (*alluxio.option.Rename*) – Options to be used when renaming a path.

Raises

- *alluxio.exceptions.NotFoundError* – If the path does not exist.
- *alluxio.exceptions.AlluxioError* – For any other exceptions thrown by Alluxio servers. Check the error status for additional details.
- *alluxio.exceptions.HTTPError* – If the underlying HTTP client library raises an error.

set_attribute(*path*, *opt=None*)
Set attributes of a path in Alluxio.

Parameters

- **path** (*str*) – The Alluxio path.
- **opt** (*alluxio.option.SetAttribute*) – Options to be used when setting attribute.

Raises

- *alluxio.exceptions.NotFoundError* – If the path does not exist.
- *alluxio.exceptions.AlluxioError* – For any other exceptions thrown by Alluxio servers. Check the error status for additional details.
- *alluxio.exceptions.HTTPError* – If the underlying HTTP client library raises an error.

unmount(*path*, *opt=None*)
Unmount an under storage that is mounted at path.

Parameters

- **path** (*str*) – The Alluxio mount point.

- **opt** (*alluxio.option.Unmount*) – Options to be used when unmounting an under storage.

Raises

- *alluxio.exceptions.AlluxioError* – For any other exceptions thrown by Alluxio servers. Check the error status for additional details.
- *alluxio.exceptions.HTTPError* – If the underlying HTTP client library raises an error.

write(*file_id*)
Creates a `Writer` for writing a file.

Parameters **file_id** (*int*) – The file ID returned by *create_file()*.

Returns The reader for reading the file as a stream.

Return type *Reader*

class alluxio.client.**Reader**(*session*, *url*)
Alluxio file reader.

The file is read as a stream; you cannot seek to a previously read section. `alluxio.Reader.close()` must be called after the reading is done.

The reader can be used as an iterator where response stream is read as chunks of bytes.

This class is used by *Client.open()*, it is not intended to be created by users directly.

All operations on the reader will raise *alluxio.exceptions.HTTPError* if the underlying HTTP client library raises errors and raise *alluxio.exceptions.AlluxioError* or its subclasses for exceptions from Alluxio.

Parameters

- **session** (`requests.Session`) –
- **url** (*str*) – The Alluxio REST URL for reading a file.

close()
Close the reader.

If the request fails, this is a no-op. Otherwise, the connection is released back into the pool. Once this method has been called, *read()* should not be called again.

read(*n=None*)
Read the file stream.

Parameters **n** (*int, optional*) – The bytes to read from the stream, if n is None, it means read the whole data stream.

Returns The data in bytes, if all data has been read, returns an empty string.

class alluxio.client.**Writer**(*session*, *url*)
Alluxio file writer.

A string or a file-like object can be written as a stream to an Alluxio file. `alluxio.Writer.close()` must be called after the writing is done.

This class is used by *Client.open()*, it is not intended to be created by users directly.

All operations on the reader will raise *alluxio.exceptions.HTTPError* if the underlying HTTP client library raises errors and raise *alluxio.exceptions.AlluxioError* or its subclasses for exceptions from Alluxio.

Parameters

- **session** (requests.Session) –
- **url** (*str*) – The Alluxio REST URL for writing a file.

close()
Close the writer.

If the request fails, this is a nop, otherwise, release the connection back to the pool. Once this method has been called, the *write()* should not be called again.

write(*data*)
Write data as a stream to the file.

The consequent calls to write will append data to the file.

Parameters **data** – data is either a string or a file-like object in python.

Returns The number of bytes that have been written.

CHAPTER 2

Client Options

Options for Alluxio Client methods.

Notes

All classes in this module have a **json** method, which converts the class into a python dict that can be encoded into a json string.

class alluxio.option.**CreateDirectory**(***kwargs*)
Options to be used in *alluxio.Client.create_directory()*.

Parameters

- **allow_exists** (*bool*) – Whether the directory can already exist.
- **mode** (*alluxio.wire.Mode*) – The directory's access mode.
- **recursive** (*bool*) – Whether to create intermediate directories along the path as necessary.
- **write_type** (*alluxio.wire.WriteType*) – Where to create the directory, e.g. in Alluxio only, in under storage only, or in both.

class alluxio.option.**CreateFile**(***kwargs*)
Options to be used in *alluxio.Client.create_file()*.

Parameters

- **block_size_bytes** (*int*) – Block size of the file in bytes.
- **location_policy_class** (*str*) – The Java class name for the location policy. If this is not specified, Alluxio will use the default value of the property key **alluxio.user.file.write.location.policy.class**.
- **mode** (*alluxio.wire.Mode*) – The file's access mode.
- **recursive** (*bool*) – If True, creates intermediate directories along the path as necessary.

- **ttl** (*int*) – The TTL (time to live) value. It identifies duration (in milliseconds) the created file should be kept around before it is automatically deleted. -1 means no TTL value is set.
- **ttl_action** (*alluxio.wire.TTLAction*) – The file action to take when its TTL expires.
- **write_type** (*alluxio.wire.WriteType*) – It can be used to decide where the file will be created, like in Alluxio only, or in both Alluxio and under storage.
- **replication_durable** (*int*) – The number of block replication for durable write.
- **replication_max** (*int*) – The maximum number of block replication.
- **replication_min** (*int*) – The minimum number of block replication.

class alluxio.option.**Delete**(***kwargs*)
Options to be used in *alluxio.Client.delete()*.

Parameters **recursive** (*bool*) – If set to true for a path to a directory, the directory and its contents will be deleted.

class alluxio.option.**Exists**
Options to be used in *alluxio.Client.exists()*.

Currently, it is an empty class, options may be added in future releases.

class alluxio.option.**Free**(***kwargs*)
Options to be used in *alluxio.Client.free()*.

Parameters **recursive** (*bool*) – If set to true for a path to a directory, the directory and its contents will be freed.

class alluxio.option.**GetStatus**
Options to be used in *alluxio.Client.get_status()*.

Currently, it is an empty class, options may be added in future releases.

class alluxio.option.**ListStatus**(***kwargs*)
Options to be used in *alluxio.Client.list_status()*.

Parameters **load_metadata_type** (*alluxio.wire.LoadMetadataType*) – The type of loading metadata, can be one of *alluxio.wire.LOAD_METADATA_TYPE_NEVER*, *alluxio.wire.LOAD_METADATA_TYPE_ONCE*, *alluxio.wire.LOAD_METADATA_TYPE_ALWAYS*, see documentation on *alluxio.wire.LoadMetadataType* for more details

class alluxio.option.**Mount**(***kwargs*)
Options to be used in *alluxio.Client.mount()*.

Parameters

- **properties** (*dict*) – A dictionary mapping property key strings to value strings.
- **read_only** (*bool*) – Whether the mount point is read-only.
- **shared** (*bool*) – Whether the mount point is shared with all Alluxio users.

class alluxio.option.**Unmount**
Options to be used in *alluxio.Client.unmount()*.

Currently, it is an empty class, options may be added in future releases.

class alluxio.option.**OpenFile**(***kwargs*)
Options to be used in *alluxio.Client.open_file()*.

Parameters

- **cache_location_policy_class** (*str*) – The Java class name for the location policy to be used when caching the opened file. If this is not specified, Alluxio will use the default value of the property key **alluxio.user.file.write.location.policy.class**.
- **max_ufs_read_concurrency** (*int*) – The maximum UFS read concurrency for one block on one Alluxio worker.
- **read_type** (*alluxio.wire.ReadType*) – The read type, like whether the file read should be cached, if this is not specified, Alluxio will use the default value of the property key **alluxio.user.file.readtype.default**.
- **ufs_read_location_policy_class** (*str*) – The Java class name for the location policy to be used when reading from under storage. If this is not specified, Alluxio will use the default value of the property key **alluxio.user.ufs.block.read.location.policy**.

class alluxio.option.**Rename**

Options to be used in *alluxio.Client.rename()*.

Currently, it is an empty class, options may be added in future releases.

class alluxio.option.**SetAttribute**(***kwargs*)

Options to be used in *alluxio.Client.set_attribute()*.

Parameters

- **owner** (*str*) – The owner of the path.
- **group** (*str*) – The group of the path.
- **mode** (*alluxio.wire.Mode*) – The access mode of the path.
- **pinned** (*bool*) – Whether the path is pinned in Alluxio, which means it should be kept in memory.
- **recursive** (*bool*) – Whether to set ACL (access control list) recursively under a directory.
- **ttl** (*int*) – The TTL (time to live) value. It identifies duration (in milliseconds) the file should be kept around before it is automatically deleted. -1 means no TTL value is set.
- **ttl_action** (*alluxio.wire.TTLAction*) – The file action to take when its TTL expires.

CHAPTER 3

Client Wire Objects

Classes in this module define the wire format of the data sent from the REST API server.

All the classes in this module have a **json** method and a **from_json** static method. The **json** method converts the class instance to a python dictionary that can be encoded into a json string. The **from_json** method decodes a json string into a class instance.

class alluxio.wire.**Bits**(*name=''*)

String representation of the access mode's bits.

Parameters **name** (*str*) – The string representation of the access mode.

Examples

The unix mode bits *wrx* can be represented as *BITS_ALL*.

Existing instances are:

- *BITS_NONE*
- *BITS_EXECUTE*
- *BITS_WRITE*
- *BITS_WRITE_EXECUTE*
- *BITS_READ*
- *BITS_READ_EXECUTE*
- *BITS_READ_WRITE*
- *BITS_ALL*

alluxio.wire.**BITS_NONE = "NONE"**

No access.

alluxio.wire.**BITS_EXECUTE = "EXECUTE"**

Execute access.

alluxio.wire.**BITS_WRITE = "WRITE"**
Write access.

alluxio.wire.**BITS_WRITE_EXECUTE = "WRITE_EXECUTE"**
Write and execute access.

alluxio.wire.**BITS_READ = "READ"**
Read access.

alluxio.wire.**BITS_READ_EXECUTE = "READ_EXECUTE"**
Read and execute access.

alluxio.wire.**BITS_READ_WRITE = "READ_WRITE"**
Read and write access.

alluxio.wire.**BITS_ALL = "ALL"**
Read, write, and execute access

class alluxio.wire.**BlockInfo**(*block_id=0*, *length=0*, *locations=[]*)
A block's information.

Parameters

- **block_id** (*int*) – Block ID.
- **length** (*int*) – Block size in bytes.
- **locations** (list of *alluxio.wire.BlockLocation*) – List of file block locations.

class alluxio.wire.**BlockLocation**(*worker_id=0*, *worker_address={"dataPort": 0, "host": "", "rpcPort": 0, "webPort": 0}*, *tier_alias=''*)
A block's location.

Parameters

- **worker_id** (*int*) – ID of the worker that contains the block.
- **worker_address** (*alluxio.wire.WorkerNetAddress*) – Address of the worker that contains the block.
- **tier_alias** (*str*) – Alias of the Alluxio storage tier that contains the block, for example, MEM, SSD, or HDD.

class alluxio.wire.**FileBlockInfo**(*block_info={"length": 0, "blockId": 0, "locations": []}*, *offset=0*, *ufs_locations=[]*)
A file block's information.

Parameters

- **block_info** (*alluxio.wire.BlockInfo*) – The block's information.
- **offset** (*int*) – The block's offset in the file.
- **ufs_locations** (*list of str*) – The under storage locations that contain this block.

class alluxio.wire.**FileInfo**(*block_ids=[]*, *block_size_bytes=0*, *cacheable=False*, *completed=False*, *creation_time_ms=0*, *last_modification_time_ms=0*, *file_block_infos=[]*, *file_id=0*, *folder=False*, *owner=''*, *group=''*, *in_memory_percentage=0*, *length=0*, *name=''*, *path=''*, *ufs_path=''*, *pinned=False*, *persisted=False*, *persistence_state=''*, *mode=0*, *mount_point=False*, *ttl=0*, *ttl_action=''*)
A file or directory's information.

Two *FileInfo* are comparable based on the attribute **name**. So a list of *FileInfo* can be sorted by python's built-in **sort** function.

Parameters

- **block_ids** (*list of int*) – List of block IDs.
- **block_size_bytes** (*int*) – Block size in bytes.
- **cacheable** (*bool*) – Whether the file can be cached in Alluxio.
- **completed** (*bool*) – Whether the file has been marked as completed.
- **creation_time_ms** (*int*) – The epoch time the file was created.
- **last_modification_time_ms** (*int*) – The epoch time the file was last modified.
- **file_block_infos** (list of *alluxio.wire.FileBlockInfo*) – List of file block information.
- **file_id** (*int*) – File ID.
- **folder** (*bool*) – Whether this is a directory.
- **owner** (*str*) – Owner of this file or directory.
- **group** (*str*) – Group of this file or directory.
- **in_memory_percentage** (*int*) – Percentage of the in memory data.
- **length** (*int*) – File size in bytes.
- **name** (*str*) – File name.
- **path** (*str*) – Absolute file path.
- **ufs_path** (*str*) – Under storage path of this file.
- **pinned** (*bool*) – Whether the file is pinned.
- **persisted** (*bool*) – Whether the file is persisted.
- **persistence_state** (`alluxio.wire.PersistenceState`) – Persistence state.
- **mode** (*int*) – Access mode of the file or directory.
- **mount_point** (*bool*) – Whether this is a mount point.
- **ttl** (*int*) – The TTL (time to live) value. It identifies duration (in milliseconds) the created file should be kept around before it is automatically deleted. -1 means no TTL value is set.
- **ttl_action** (*alluxio.wire.TTLAction*) – The file action to take when its TTL expires.

class alluxio.wire.**LoadMetadataType**(*name=''*)

The way to load metadata.

This can be one of the following, see their documentation for details:

- *LOAD_METADATA_TYPE_NEVER*
- *LOAD_METADATA_TYPE_ONCE*
- *LOAD_METADATA_TYPE_ALWAYS*

Parameters **name** (*str*) – The string representation of the way to load metadata.

alluxio.wire.**LOAD_METADATA_TYPE_NEVER = "Never"**

Never load metadata.

alluxio.wire.**LOAD_METADATA_TYPE_ONCE = "Once"**
Load metadata only at the first time of listing status on a directory.

alluxio.wire.**LOAD_METADATA_TYPE_ALWAYS = "Always"**
Always load metadata when listing status on a directory.

class alluxio.wire.**Mode**(*owner_bits=""*, *group_bits=""*, *other_bits=""*)
A file's access mode.

Parameters

- **owner_bits** (*alluxio.wire.Bits*) – Access mode of the file's owner.
- **group_bits** (*alluxio.wire.Bits*) – Access mode of the users in the file's group.
- **other_bits** (*alluxio.wire.Bits*) – Access mode of others who are neither the owner nor in the group.

class alluxio.wire.**ReadType**(*name=''*)
Convenience modes for commonly used read types.

This can be one of the following, see their documentation for details:

- *READ_TYPE_NO_CACHE*
- *READ_TYPE_CACHE*
- *READ_TYPE_CACHE_PROMOTE*

Parameters **name** (*str*) – The string representation of the read type.

alluxio.wire.**READ_TYPE_NO_CACHE = "NO_CACHE"**
Read the file and skip Alluxio storage. This read type will not cause any data migration or eviction in Alluxio storage.

alluxio.wire.**READ_TYPE_CACHE = "CACHE"**
Read the file and cache it in the highest tier of a local worker. This read type will not move data between tiers of Alluxio Storage. Users should use *READ_TYPE_CACHE_PROMOTE* for more optimized performance with tiered storage.

alluxio.wire.**READ_TYPE_CACHE_PROMOTE = "CACHE_PROMOTE"**
Read the file and cache it in a local worker. Additionally, if the file was in Alluxio storage, it will be promoted to the top storage layer.

class alluxio.wire.**TTLAction**(*name=''*)
Represent the file action to take when its TTL expires.

This can be one of the following, see their documentation for details:

- *TTL_ACTION_DELETE*
- *TTL_ACTION_FREE*

Parameters **name** (*str*) – The string representation of the read type.

alluxio.wire.**TTL_ACTION_DELETE = "DELETE"**
Represents the action of deleting a path.

alluxio.wire.**TTL_ACTION_FREE = "FREE"**
Represents the action of freeing a path.

class alluxio.wire.**WorkerNetAddress**(*host=''*, *rpc_port=0*, *data_port=0*, *web_port=0*)
Worker network address.

Parameters

- **host** (*str*) – Worker's hostname.
- **rpc_port** (*int*) – Port of the worker's RPC server.
- **data_port** (*int*) – Port of the worker's data server.
- **web_port** (*int*) – Port of the worker's web server.

class alluxio.wire.**WriteType**(*name=''*)

Write types for creating a file.

This can be one of the following, see their documentation for details:

- *WRITE_TYPE_MUST_CACHE*
- *WRITE_TYPE_CACHE_THROUGH*
- *WRITE_TYPE_THROUGH*
- *WRITE_TYPE_ASYNC_THROUGH*

Parameters **name** (*str*) – The string representation of the write type.

alluxio.wire.**WRITE_TYPE_MUST_CACHE = "MUST_CACHE"**

Write the file, guaranteeing the data is written to Alluxio storage or failing the operation. The data will be written to the highest tier in a worker's storage. Data will not be persisted to the under storage.

alluxio.wire.**WRITE_TYPE_CACHE_THROUGH = "CACHE_THROUGH"**

Write the file synchronously to the under storage, and also try to write to the highest tier in a worker's Alluxio storage.

alluxio.wire.**WRITE_TYPE_THROUGH = "THROUGH"**

Write the file synchronously to the under storage, skipping Alluxio storage.

alluxio.wire.**WRITE_TYPE_ASYNC_THROUGH = "ASYNC_THROUGH"**

Write the file asynchronously to the under storage.

CHAPTER 4

Client Exceptions

Alluxio exceptions.

All exceptions thrown by *alluxio.Client*, alluxio.Reader, and alluxio.Writer are one of the following exceptions:

- *AlluxioError*
- subclasses of *AlluxioError*
- *HTTPError*
- Python built-in exceptions

Exceptions raised by the requests library are wrapped by *HTTPError*.

class alluxio.exceptions.**Status**
A class representing RPC status codes.

The definitions are from https://github.com/Alluxio/alluxio/blob/master/core/common/src/main/java/alluxio/exception/status/Status.java.

exception alluxio.exceptions.**AlluxioError**(*status*, *message*)
Base class for all Alluxio exceptions.

Parameters

- **status** (*str*) – The status defined in *Status*.
- **message** (*str*) – The error message.

exception alluxio.exceptions.**AbortedError**(*message*)
Exception indicating that the operation was aborted, typically due to a concurrency issue like sequencer check failures, transaction aborts, etc.

See litmus test in FailedPreconditionException for deciding between FailedPreconditionException, AbortedException, and UnavailableException.

Parameters **message** (*str*) – The error message.

exception alluxio.exceptions.**AlreadyExistsError**(*message*)
Exception indicating that an attempt to create an entity failed because one already exists.

Parameters **message** (*str*) – The error message.

exception alluxio.exceptions.**CanceledError**(*message*)
Exception indicating that an operation was cancelled (typically by the caller).

Parameters **message** (*str*) – The error message.

exception alluxio.exceptions.**DataLossError**(*message*)
Exception indicating unrecoverable data loss or corruption.

Parameters **message** (*str*) – The error message.

exception alluxio.exceptions.**DeadlineExceededError**(*message*)
Exception indicating that an operation expired before completion. For operations that change the state of the system, this exception may be thrown even if the operation has completed successfully. For example, a successful response from a server could have been delayed long enough for the deadline to expire.

Parameters **message** (*str*) – The error message.

exception alluxio.exceptions.**FailedPreconditionError**(*message*)
Exception indicating that operation was rejected because the system is not in a state required for the operation's execution. For example, directory to be deleted may be non-empty, an rmdir operation is applied to a non-directory, etc.

A litmus test that may help a service implementor in deciding between `FailedPreconditionException`, `AbortedException`, and `UnavailableException`:

1. Use UnavailableException if the client can retry the failed call.
2. Use AbortedException if the client should retry at a higher-level (e.g., restarting a read-modify-write sequence).
3. Use FailedPreconditionException if the client should not retry until the system state has been explicitly fixed. E.g., if an "rmdir" fails because the directory is non-empty, FailedPreconditionException should be thrown since the client should not retry unless they have first fixed up the directory by deleting files from it.
4. Use FailedPreconditionException if the client performs conditional REST Get/Update/Delete on a resource and the resource on the server does not match the condition. E.g. conflicting read-modify-write on the same resource.

Parameters **message** (*str*) – The error message.

exception alluxio.exceptions.**InternalError**(*message*)
Exception representing an internal error. This means some invariant expected by the underlying system has been broken. If you see one of these errors, something is very broken.

Parameters **message** (*str*) – The error message.

exception alluxio.exceptions.**InvalidArgumentError**(*message*)
Exception indicating that a client specified an invalid argument. Note that this differs from `FailedPreconditionException`. It indicates arguments that are problematic regardless of the state of the system (e.g., a malformed file name).

Parameters **message** (*str*) – The error message.

exception alluxio.exceptions.**NotFoundError**(*message*)
Exception indicating that some requested entity (e.g., file or directory) was not found.

Parameters **message** (*str*) – The error message.

exception alluxio.exceptions.**OutOfRangeError**(*message*)
Exception indicating that and operation was attempted past the valid range. E.g., seeking or reading past end of file.

Unlike `InvalidArgumentException`, this error indicates a problem that may be fixed if the system state changes. For example, a 32-bit file system will generate `InvalidArgumentException` if asked to read at an offset that is not in the range [0,2^32-1], but it will generate `OutOfRangeException` if asked to read from an offset past the current file size.

There is a fair bit of overlap between `FailedPreconditionException` and `OutOfRangeException`. We recommend using `OutOfRangeException` (the more specific error) when it applies so that callers who are iterating through a space can easily look for an `OutOfRangeException` to detect when they are done.

Parameters **message** (*str*) – The error message.

exception alluxio.exceptions.**PermissionDeniedError**(*message*)
Exception indicating that the caller does not have permission to execute the specified operation. It must not be used for rejections caused by exhausting some resource (use `ResourceExhaustedException` instead for those exceptions). It must not be used if the caller cannot be identified (use `UnauthenticatedException` instead for those exceptions).

Parameters **message** (*str*) – The error message.

exception alluxio.exceptions.**ResourceExhaustedError**(*message*)
Exception indicating that some resource has been exhausted, perhaps a per-user quota, or perhaps the entire file system is out of space.

Parameters **message** (*str*) – The error message.

exception alluxio.exceptions.**UnauthenticatedError**(*message*)
Exception indicating that the request does not have valid authentication credentials for the operation.

Parameters **message** (*str*) – The error message.

exception alluxio.exceptions.**UnavailableError**(*message*)
Exception indicating that the service is currently unavailable.

This is a most likely a transient condition and may be corrected by retrying with a backoff.

See litmus test in `FailedPreconditionException` for deciding between `FailedPreconditionException`, `AbortedException`, and `UnavailableException`.

Parameters **message** (*str*) – The error message.

exception alluxio.exceptions.**UnimplementedError**(*message*)
Exception indicating that an operation is not implemented, or not supported, or not enabled.

Parameters **message** (*str*) – The error message.

exception alluxio.exceptions.**UnknownError**(*message*)
Exception representing an unknown error. An example of where this exception may be thrown is if a Status value received from another address space belongs to an error-space that is not known in this address space. Also errors raised by APIs that do not return enough error information may be converted to this error.

Parameters **message** (*str*) – The error message.

alluxio.exceptions.**new_alluxio_exception**(*status*, *message*)
Creates the appropriate exception for status.

If status is not defined in *Status*, then creates a general *AlluxioError*.

Parameters

- **status** (*str*) – The status defined in *Status*.

- **message** (*str*) – The error message.

exception alluxio.exceptions.**HTTPError**

Any error raised by the underlying HTTP client library will be wrapped by this error.

Python Module Index

a

Index

N

O

P

R

S

T

U

W