PyXenon Documentation

Release 2.2.0

Johan Hidding, Stefan Verhoeven

Jun 13, 2018

Contents

CHAPTER 1

Quick Start

We like to test Xenon against a Docker image: nlesc/xenon-slurm. If you have docker all setup, you can run this image as follows:

```
$ docker pull nlesc/xenon-slurm
...
$ docker run --detach --publish 10022:22 nlesc/xenon-slurm
```

Try logging onto this image by *ssh*, to make sure everything works. The username is *xenon*, the password *javagat*:

```
$ ssh localhost -p 10022 -l xenon
xenon@localhost's password: <javagat>
$ exit
Connection to localhost closed.
```

1.1 Starting the server

To get anything done in PyXenon, we need to start the GRPC server:

```
import xenon

xenon.init()
```

1.2 Writing to a remote filesystem

Next, let's try to copy a file to the container. We need credentials to access anything on the remote side.

```
from xenon import PasswordCredential, FileSystem

credential = PasswordCredential(
```

(continues on next page)

(continued from previous page)

```
    username='xenon',
    password='javagat')

remotefs = FileSystem.create(
    'sftp', location='localhost:10022',
    password_credential=credential)
```

We can write to a file by streaming. The second argument to `write_to_file()` should be an iterable. It will be read in a separate thread, so it is allowed to be blocking. Here we'll do nothing so fancy:

```
from xenon import Path

target = Path('hello.sh')

if remotefs.exists(target):
    remotefs.delete(target)

remotefs.write_to_file(
    target,
    [b'#!/bin/sh\n',
     b'echo "Hello, World!"\n'])
```

1.3 Running a script

The remote machine runs a SLURM job scheduler. We describe a job in a `JobDescription` object. This seems a bit long-winded, but in practice you'll be reusing the descriptions a lot.

```
from xenon import Scheduler, JobDescription

scheduler = Scheduler.create(
    adaptor='slurm',
    location='ssh://localhost:10022',
    password_credential=credential)

job_description = JobDescription(
    executable='/bin/sh',
    arguments=['hello.sh'],
    stdout='result.txt')

job = scheduler.submit_batch_job(job_description)

state = scheduler.wait_until_done(job)
print(state)
```

1.4 Retrieving the result

Just as we can write data by sending an iterable, we can read data from a file and recieve a generator yielding bytes objects. Here we realize the transfer by joining the data chunks into a string:

```
text = ''.join(chunk.decode() for chunk in
    remotefs.read_from_file(Path('result.txt')))
print(text)
```

CHAPTER 2

Advanced: Streaming & Interactive jobs

In several cases it is desireable to stream data from/to interactive jobs as well as data to a remote filesystem. The GRPC API has build-in support for asynchronous streaming through many simultaneous requests. In Python this API is exposed in terms of generators.

2.1 Example: an online job

In this example we'll show how to obtain bi-directional communication with an online job. An online job is started with `Scheduler.submit_online_job()`.

2.1.1 Streaming input, a.k.a. The Halting Problem

We need to stream input to the online job. In the *Quick Start*, we saw that we could send data to a stream by simply giving a list of bytes objects. Here we aim a bit more advanced to play a kind of real-time ping-pong with a remote process. We need to provide *PyXenon* with an generator that pulls its messages from a queue. The GRPC module ensures that this generator is being run asynchonously from the main thread.

The tricky part is that we need to be able to tell the generator when the work is done and no more input is to be expected. We could have it recieve strings and make it check for end-of-file messages in some way, but in essence we'll always have to define a little protocol to deal with the finiteness of the generator's life. To make this explicit we define a little 2-tuple micro-language:

message	action
`('msg', <value: string>)`	`yield value.encode()`
`('end', None)`	`return`

Implementing this:

```
from queue import Queue

def make_input_stream():
    input_queue = Queue()

    def input_stream():
        while True:
            cmd, value = input_queue.get()
            if cmd == 'end':
                input_queue.task_done()
                return
            elif cmd == 'msg':
                yield value.encode()
                input_queue.task_done()

    return input_queue, input_stream
```

2.1.2 Reading output

The return-value of `submit_online_job()` is an iterator yielding objects of type *SubmitOnlineJobResponse*. These objects have a `stdout` field containing (binary) data that the job wrote to standard output, as well as a `stderr` field containing data written to standard error. For any message either field may be empty or not. In this example we're only interested in data from `stdout`:

```
def get_stdout(stream):
    return stream.next().stdout.decode()
```

2.1.3 The "remote" script

For the purpose of this example, we have defined a small Python `rot13` program:

Listing 1: rot13.py

```
import codecs

try:
    while True:
        line = input()
        print(codecs.encode(line, 'rot_13'))

except EOFError:
    pass
```

2.1.4 Defining the job

Online job descriptions are the same as normal job descriptions.

```
# our input lines
input_lines = [
    "Zlfgvp aboyr tnf,",
    "Urnil lrg syrrgvat sebz tenfc,",
    "Oyhr yvxr oheavat vpr."
```

(continues on next page)

(continued from previous page)

```
]

# the job description, make sure you run the script from the examples
# directory!
job_description = xenon.JobDescription(
    executable='python',
    arguments=['rot13.py'],
    queue_name='multi')
```

2.1.5 Putting it together

The rest is history.

```
import xenon

# start the xenon-grpc server
xenon.init()

# on the local adaptor
with xenon.Scheduler.create(adaptor='local') as scheduler:
    input_queue, input_stream = make_input_stream()

    # submit an interactive job, this gets us the job-id and a stream
    # yielding job output from stdout and stderr.
    job, output_stream = scheduler.submit_interactive_job(
        description=job_description, stdin_stream=input_stream())

    # next we feed the input_queue with messages
    try:
        for line in input_lines:
            print(" [sending]    " + line)
            input_queue.put(('msg', line + '\n'))
            msg = get_stdout(output_stream)
            print("[received]    " + msg)

    # make sure to close our end whatever may happen
    finally:
        input_queue.put(('end', None))
        input_queue.join()

    scheduler.wait_until_done(job)
```

2.2 Protocol definitions

It can be instructive to see what the GRPC protocol with respect to interactive jobs looks like.

```
message SubmitInteractiveJobRequest {
    Scheduler scheduler = 1;
    JobDescription description = 2;
    bytes stdin = 3;
}
```

(continues on next page)

(continued from previous page)

```
message SubmitInteractiveJobResponse {
    Job job = 1;
    bytes stdout = 2;
    bytes stderr = 3;
}

service SchedulerService {
    rpc submitInteractiveJob(
            stream SubmitInteractiveJobRequest)
        returns (stream SubmitInteractiveJobResponse) {}
}
```

In *PyXenon* the remote procedure call `submitInteractiveJob` is wrapped to the method `submit_interactive_job()` of the `Scheduler` class. Note that the `SubmitInteractiveJobRequest` specifies (next to the scheduler, which is obtained from `self` in the method call) the job description and `bytes` for standard input. Requests of this type are streamed. This means that GRPC expects to get an iterator of `SubmitInteractiveJobRequest` objets.

The *PyXenon* `submit_interactive_job()` method separates the job-description and input-stream arguments. Sending the `scheduler` and `description` fields in the first request, followed up by a sequence of requests where only the `stdin` field is specified. This latter sequence is yielded from the `stdin_stream` argument.

Similarly, the first item in the output stream is guaranteed to only contain the job-id, this first item is available immediately. Subsequent calls to `next(output_stream)` will block until output is available. The `submit_interactive_job()` method takes the first item of the iterator, and extracts the job-id. The user recieves a tuple with the extracted job-id and the iterator.

CHAPTER 3

Adaptors

This section contains the adaptor documentation which is generated from the information provided by the adaptors themselves.

Contents

3.1 File System

Note: Supported property names should be prefixed with

`"xenon.adaptors.filesystems"`. We've left this prefix out to improve readability of the tables.

3.1.1 File

This is the local file adaptor that implements file functionality for local access.

field	value
supports_third_party_copy	False
can_create_symboliclinks	True
can_read_symboliclinks	True
is_connectionless	True
supported_credentials	*DefaultCredential*
can_append	True
supports_reading_posix_permissions	True
supports_setting_posix_permissions	True
supports_rename	True
needs_size_beforehand	False

location string:

- *(null)*
- *(empty string)*
- *[/workdir]*
- *driveletter:[/workdir]*

supported properties:

name	description	data_type	default
file.bufferSize	The buffer size to use when copying files (in bytes).	size	*64K*

3.1.2 Ftp

The FTP adaptor implements file access on remote ftp servers.

field	value
supports_third_party_copy	False
can_create_symboliclinks	False
can_read_symboliclinks	True
is_connectionless	False
supported_credentials	*DefaultCredential*, *PasswordCredential*
can_append	True
supports_reading_posix_permissions	True
supports_setting_posix_permissions	False
supports_rename	True
needs_size_beforehand	False

location string:

- *host[:port][/workdir]*

supported properties:

name	description	data_type	default
ftp.bufferSize	The buffer size to use when copying files (in bytes).	size	*64K*

3.1.3 Sftp

The SFTP adaptor implements all file access functionality to remote SFTP servers

field	value
supports_third_party_copy	False
can_create_symboliclinks	True
can_read_symboliclinks	True
is_connectionless	False
supported_credentials	*DefaultCredential*, *CertificateCredential*, *PasswordCredential*, *CredentialMap*
can_append	True
supports_reading_posix_permissions	True
supports_setting_posix_permissions	True
supports_rename	True
needs_size_beforehand	False

location string:

- *host[:port][/workdir]*

supported properties:

name	description	data_type	default
sftp.strictHostKeyChecking	Enable strict host key checking.	boolean	*true*
sftp.loadKnownHosts	Load the standard known_hosts file.	boolean	*true*
sftp.loadSshConfig	Load the OpenSSH config file.	boolean	*true*
sftp.agent	Use a (local) ssh-agent.	boolean	*false*
sftp.agentForwarding	Use ssh-agent forwarding when setting up a connection.	boolean	*false*
sftp.connection.timeout	The timeout for creating and authenticating connections (in milliseconds).	natural	*10000*
sftp.bufferSize	The buffer size to use when copying files (in bytes).	size	*64K*

3.1.4 Webdav

The webdav file adaptor implements file access to remote webdav servers.

field	value
supports_third_party_copy	False
can_create_symboliclinks	False
can_read_symboliclinks	False
is_connectionless	True
supported_credentials	*DefaultCredential*, *PasswordCredential*
can_append	False
supports_reading_posix_permissions	False
supports_setting_posix_permissions	False
supports_rename	True
needs_size_beforehand	False

location string:

- *http://host[:port][/workdir]*
- *https://host[:port][/workdir]*

supported properties:

name	description	data_type	default
webdav.bufferSize	The buffer size to use when copying files (in bytes).	size	*64K*

3.1.5 S3

The JClouds adaptor uses Apache JClouds to talk to s3 and others

field	value
supports_third_party_copy	False
can_create_symboliclinks	False
can_read_symboliclinks	False
is_connectionless	True
supported_credentials	*PasswordCredential*
can_append	False
supports_reading_posix_permissions	False
supports_setting_posix_permissions	False
supports_rename	False
needs_size_beforehand	True

location string:

- *[http://host[:port]]/bucketname[/workdir]*

supported properties:

name	description	data_type	default
s3.bufferSize	The buffer size to use when copying files (in bytes).	size	*64K*

3.1.6 Hdfs

Adaptor for the Apache Hadoop file system

field	value
supports_third_party_copy	False
can_create_symboliclinks	False
can_read_symboliclinks	False
is_connectionless	False
supported_credentials	*DefaultCredential*, *PasswordCredential*, *KeytabCredential*
can_append	True
supports_reading_posix_permissions	False
supports_setting_posix_permissions	False
supports_rename	True
needs_size_beforehand	False

location string:

- *hdfs://host[:port]*

supported properties:

name	description	data_type	default
hdfs.bufferSize	The buffer size to use when copying files (in bytes).	size	*64K*
hdfs.hadoopSettingsFile	The path to the file with the hadoop settings, i.e. "/home/xenon/core-site.xml".	string	(empty)

3.2 Scheduler

Note: Supported property names should be prefixed with

`"xenon.adaptors.schedulers"`. We've left this prefix out to improve readability of the tables.

3.2.1 Local

The local jobs adaptor implements all functionality by emulating a local queue.

field	value
is_embedded	True
supports_interactive	True
supports_batch	True
uses_file_system	True
supported_credentials	*DefaultCredential*

location string:

- *[/workdir]*

supported properties:

name	description	data_type	default
local.queue.pollingDelay	The polling delay for monitoring running jobs (in milliseconds).	long	*1000*
local.queue.multi.maxConcurrentJobs	The maximum number of concurrent jobs in the multiq.	integer	*4*

3.2.2 Ssh

The SSH job adaptor implements all functionality to start jobs on ssh servers.

field	value
is_embedded	True
supports_interactive	True
supports_batch	True
uses_file_system	True
supported_credentials	*DefaultCredential*, *CertificateCredential*, *PasswordCredential*, *CredentialMap*

location string:

- *host[:port][/workdir][via:otherhost[:port]]**

supported properties:

name	description	data_type	default
ssh.strictHostKeyChecking	Enable strict host key checking.	boolean	*true*
ssh.loadKnownHosts	Load the standard known_hosts file.	boolean	*true*
ssh.loadSshConfig	Load the OpenSSH config file.	boolean	*true*
ssh.agent	Use a (local) ssh-agent.	boolean	*false*
ssh.agentForwarding	Use ssh-agent forwarding	boolean	*false*
ssh.timeout	The timeout for the connection setup and authetication (in milliseconds).	long	*10000*
ssh.queue.pollingDelay	The polling delay for monitoring running jobs (in milliseconds).	long	*1000*
ssh.queue.multi.maxConcurrentJobs	The maximum number of concurrent jobs in the multiq..	integer	*4*

3.2.3 Gridengine

The SGE Adaptor submits jobs to a (Sun/Oracle/Univa) Grid Engine scheduler. This adaptor uses either the local or the ssh scheduler adaptor to run commands on the machine running Grid Engine, and the file or the stfp filesystem adaptor to gain access to the filesystem of that machine.

field	value
is_embedded	False
supports_interactive	False
supports_batch	True
uses_file_system	True
supported_credentials	*DefaultCredential*, *CertificateCredential*, *PasswordCredential*, *CredentialMap*

location string:

- *local://[/workdir]*
- *ssh://host[:port][/workdir][via:otherhost[:port]]**

supported properties:

name	description	data_type	default
gridengine.ignore.version	Skip version check is skipped when connecting to remote machines. WARNING: it is not recommended to use this setting in production environments!	boolean	*false*
gridengine.accounting.grace.time	Number of milliseconds a job is allowed to take going from the queue to the qacct output.	long	*60000*
gridengine.poll.delay	Number of milliseconds between polling the status of a job.	long	*1000*
ssh.strictHostKeyChecking	Enable strict host key checking.	boolean	*true*
ssh.loadKnownHosts	Load the standard known_hosts file.	boolean	*true*
ssh.loadSshConfig	Load the OpenSSH config file.	boolean	*true*
ssh.agent	Use a (local) ssh-agent.	boolean	*false*
ssh.agentForwarding	Use ssh-agent forwarding	boolean	*false*
ssh.timeout	The timeout for the connection setup and authetication (in milliseconds).	long	*10000*
ssh.queue.pollingDelay	The polling delay for monitoring running jobs (in milliseconds).	long	*1000*
ssh.queue.multi.maxConcurrentJobs	The maximum number of concurrent jobs in the multiq..	integer	*4*
local.queue.pollingDelay	The polling delay for monitoring running jobs (in milliseconds).	long	*1000*
local.queue.multi.maxConcurrentJobs	The maximum number of concurrent jobs in the multiq.	integer	*4*

3.2.4 Slurm

The Slurm Adaptor submits jobs to a Slurm scheduler. This adaptor uses either the local or the ssh scheduler adaptor to run commands on the machine running Slurm, and the file or the stfp filesystem adaptor to gain access to the filesystem of that machine.

field	value
is_embedded	False
supports_interactive	True
supports_batch	True
uses_file_system	True
supported_credentials	*DefaultCredential*, *CertificateCredential*, *PasswordCredential*, *CredentialMap*

location string:

- *local://[/workdir]*
- *ssh://host[:port][/workdir][via:otherhost[:port]]**

supported properties:

name	description	data_type	default
slurm.disable.accounting.usage	Do not use accounting info of slurm, even when available. Mostly for testing purposes	boolean	*false*
slurm.poll.delay	Number of milliseconds between polling the status of a job.	long	*1000*
ssh.strictHostKeyChecking	Enable strict host key checking.	boolean	*true*
ssh.loadKnownHosts	Load the standard known_hosts file.	boolean	*true*
ssh.loadSshConfig	Load the OpenSSH config file.	boolean	*true*
ssh.agent	Use a (local) ssh-agent.	boolean	*false*
ssh.agentForwarding	Use ssh-agent forwarding	boolean	*false*
ssh.timeout	The timeout for the connection setup and authetication (in milliseconds).	long	*10000*
ssh.queue.pollingDelay	The polling delay for monitoring running jobs (in milliseconds).	long	*1000*
ssh.queue.multi.maxConcurrentJobs	The maximum number of concurrent jobs in the multiq..	integer	*4*
local.queue.pollingDelay	The polling delay for monitoring running jobs (in milliseconds).	long	*1000*
local.queue.multi.maxConcurrentJobs	The maximum number of concurrent jobs in the multiq.	integer	*4*

3.2.5 Torque

The Torque Adaptor submits jobs to a TORQUE batch system. This adaptor uses either the local or the ssh scheduler adaptor to run commands on the machine running TORQUE, and the file or the stfp filesystem adaptor to gain access to the filesystem of that machine.

field	value
is_embedded	False
supports_interactive	False
supports_batch	True
uses_file_system	True
supported_credentials	*DefaultCredential*, *CertificateCredential*, *PasswordCredential*, *CredentialMap*

location string:

- *local:///[/workdir]*
- *ssh://host[:port][/workdir][via:otherhost[:port]]**

supported properties:

name	description	data_type	default
torque.ignore.version	Skip version check is skipped when connecting to remote machines. WARNING: it is not recommended to use this setting in production environments!	boolean	*false*
torque.accounting.gracetime	Number of milliseconds a job is allowed to take going from the queue to the accinfo output.	long	*60000*
torque.poll.delay	Number of milliseconds between polling the status of a job.	long	*1000*
ssh.strictHostKeyChecking	Enable strict host key checking.	boolean	*true*
ssh.loadKnownHosts	Load the standard known_hosts file.	boolean	*true*
ssh.loadSshConfig	Load the OpenSSH config file.	boolean	*true*
ssh.agent	Use a (local) ssh-agent.	boolean	*false*
ssh.agentForwarding	Use ssh-agent forwarding	boolean	*false*
ssh.timeout	The timeout for the connection setup and authetication (in milliseconds).	long	*10000*
ssh.queue.pollingDelay	The polling delay for monitoring running jobs (in milliseconds).	long	*1000*
ssh.queue.multi.maxConcurrentJobs	The maximum number of concurrent jobs in the multiq..	integer	*4*
local.queue.pollingDelay	The polling delay for monitoring running jobs (in milliseconds).	long	*1000*
local.queue.multi.maxConcurrentJobs	The maximum number of concurrent jobs in the multiq.	integer	*4*

CHAPTER 4

API

Contents

4.1 The Server

xenon.**init**(*port=None*, *do_not_exit=False*, *disable_tls=False*, *log_level='WARNING'*)

Start the Xenon GRPC server on the specified port, or, if a service is already running on that port, connect to that.

If no port is given, a random port is selected. This means that, by default, every python instance will start its own instance of a xenon-grpc process.

Parameters

- **port** – the port number
- **do_not_exit** – by default the GRPC server is shut down after Python exits (through the *atexit* module), setting this value to *True* will prevent that from happening.

4.2 File Systems

class xenon.**FileSystem**(*service*, *wrapped*)
The Xenon *FileSystem* subsystem.

Variables **id** (*string*) – id

append_to_file(*path*, *data_stream*)
Open an existing file and return an OutputStream to append data to this file.

cancel(*copy_operation=None*)
Cancel a copy operation.

Parameters **copy_operation** (*xenon.CopyOperation*) – copy_operation

close()
Close this filestem Any pending/running copy operations of this filestystem will be terminated Will also forget this filesystem

copy(*source=None*, *destination_filesystem=None*, *destination=None*, *mode=None*, *recursive=None*)
Asynchronously Copy an existing source path to a target path on a different file system.

Parameters

- **source** (xenon.Path) – source
- **destination_filesystem** (xenon.FileSystem) – destination_filesystem
- **destination** (xenon.Path) – destination
- **mode** (*xenon.CopyRequest.CopyMode*) – mode
- **recursive** (*bool*) – recursive

classmethod create(*adaptor=None*, *location=None*, *properties=None*, *certificate_credential=None*, *password_credential=None*, *default_credential=None*, *credential_map=None*, *keytab_credential=None*)
Create a new FileSystem using the adaptor that connects to a data store at location using the credentials to get access.

Parameters

- **adaptor** (*string*) – adaptor
- **location** (*string*) – location
- **properties** (*map<string, string>*) – properties
- **certificate_credential** (xenon.CertificateCredential) – certificate_credential
- **password_credential** (xenon.PasswordCredential) – password_credential
- **default_credential** (*xenon.DefaultCredential*) – default_credential
- **credential_map** (*xenon.CredentialMap*) – credential_map
- **keytab_credential** (*xenon.KeytabCredential*) – keytab_credential

create_directories(*path=None*)
Creates a new directory, including parent directories, failing if the directory already exists.

Parameters **path** (xenon.Path) – path

create_directory(*path=None*)
Creates a new directory, failing if the directory already exists.

Parameters **path** (xenon.Path) – path

create_file(*path=None*)
Creates a new empty file, failing if the file already exists.

Parameters **path** (xenon.Path) – path

create_symbolic_link(*link=None*, *target=None*)
Creates a new symbolic link, failing if the link already exists

Parameters

- **link** (xenon.Path) – link
- **target** (xenon.Path) – target

delete(*path=None*, *recursive=None*)
Deletes an existing path.

Parameters

- **path** (xenon.Path) – path
- **recursive** (*bool*) – recursive

exists(*path=None*)
Tests if a path exists.

Parameters **path** (xenon.Path) – path

classmethod get_adaptor_description(*name=None*)
Gives the description of the adaptor with the given name.

Parameters **name** (*string*) – name

classmethod get_adaptor_descriptions()
Gives a list of the descriptions of the available adaptors.

get_adaptor_name()
Get the name of the adaptor that created this FileSystem.

classmethod get_adaptor_names()
Gives a list names of the available adaptors.

get_attributes(*path=None*)
Get the PathAttributes of an existing path.

Parameters **path** (xenon.Path) – path

get_path_separator()
Get the path separator used by this file system.

get_status(*copy_operation=None*)
Retrieve the status of an copy.

Parameters **copy_operation** (*xenon.CopyOperation*) – copy_operation

get_working_directory()
Get the current working directory of this file system.

is_open()
Return if the connection to the FileSystem is open.

list(*dir=None*, *recursive=None*)
List all entries in the directory dir.

Parameters

- **dir** (xenon.Path) – dir
- **recursive** (*bool*) – recursive

classmethod list_file_systems()
List the created filesystems Specific to grpc, not part of Xenon library

classmethod local_file_systems()
Returns filesystems for all local drives Not part of FileSystem class in Xenon library In Xenon library available as LocalFileSystemUtils.getLocalFileSystems()

read_from_file(*path=None*)
Open an existing file and return an InputStream to read from this file.

Parameters **path** (xenon.Path) – path

read_symbolic_link(*path=None*)
Reads the target of a symbolic link

Parameters **path** (xenon.Path) – path

rename(*source=None*, *target=None*)
Rename an existing source path to a non-existing target path

Parameters

- **source** (xenon.Path) – source
- **target** (xenon.Path) – target

set_posix_file_permissions(*path=None*, *permissions=None*)
Sets the POSIX permissions of a path

Parameters

- **path** (xenon.Path) – path
- **permissions** (xenon.PosixFilePermission) – permissions

set_working_directory(*path=None*)
Set the current working directory of this file system to directory.

Parameters **path** (xenon.Path) – path

wait_until_done(*copy_operation=None*, *timeout=None*)
Wait until a copy operation is done or until a timeout expires.

Parameters

- **copy_operation** (*xenon.CopyOperation*) – copy_operation
- **timeout** (*uint64*) – timeout

write_to_file(*path*, *data_stream*)
Open a file and return an OutputStream to write to this file. In Xenon library if request is missing size field then FileSystem.writeToFile(Path file) is used else FileSystem.writeToFile(Path path, long size) is used

class xenon.**Path**(*path*)
Wrapper around `PurePosixPath` form the `pathlib` module. This class reveals a string representation of the underlying path object to GRPC. You may use this class like a *pathlib.PurePosixPath*, including using it as an argument to *open* calls as it derives from *os.PathLike* (Python > 3.6). For more information see the Python documentation on pathlib.

is_hidden()
Checks if a file is hidden. Just compares the first character in the filename with '.'.

4.2.1 Message classes

class xenon.**PosixFilePermission**
An enumeration.

GROUP_EXECUTE = 6

GROUP_READ = 4

GROUP_WRITE = 5

NONE = 0

OTHERS_EXECUTE = 9

OTHERS_READ = 7

OTHERS_WRITE = 8

OWNER_EXECUTE = 3

OWNER_READ = 1

OWNER_WRITE = 2

class xenon.**CopyMode**
An enumeration.

CREATE = 0

IGNORE = 2

REPLACE = 1

class xenon.**CopyStatus**(*service*, *wrapped*)
Status of a copy operation.

Variables

- **copy_operation** (*xenon.CopyOperation*) – copy_operation
- **bytes_copied** (*uint64*) – bytes_copied
- **bytes_to_copy** (*uint64*) – bytes_to_copy
- **done** (*bool*) – done
- **running** (*bool*) – running
- **state** (*string*) – state
- **error_message** (*string*) – error_message
- ***error_type*** (xenon.CopyStatus.ErrorType) – error_type

class ErrorType
An enumeration.

ALREADY_EXISTS = 3

CANCELLED = 2

NONE = 0

NOT_CONNECTED = 4

NOT_FOUND = 1

XENON = 5

error_type

4.3 Schedulers

class xenon.**Scheduler**(*service*, *wrapped*)
The Xenon Schedulers subsystem.

Variables **id** (*string*) – id

cancel_job(*job=None*)
Cancel a job

Parameters **job** (xenon.Job) – job

close()
Close this Scheduler. If scheduler is embedded then any pending/running jobs will be killed Will also forget this scheduler

classmethod create(*adaptor=None*, *location=None*, *properties=None*, *certificate_credential=None*, *password_credential=None*, *default_credential=None*, *credential_map=None*, *keytab_credential=None*)
Create a new Scheduler using the adaptor connecting to the location using credentials to get access.

Parameters

- **adaptor** (*string*) – adaptor
- **location** (*string*) – location
- **properties** (*map<string, string>*) – properties
- **certificate_credential** (xenon.CertificateCredential) – certificate_credential
- **password_credential** (xenon.PasswordCredential) – password_credential
- **default_credential** (*xenon.DefaultCredential*) – default_credential
- **credential_map** (*xenon.CredentialMap*) – credential_map
- **keytab_credential** (*xenon.KeytabCredential*) – keytab_credential

classmethod get_adaptor_description(*name=None*)
Gives the description of the adaptor with the given name.

Parameters **name** (*string*) – name

classmethod get_adaptor_descriptions()
Gives a list of the descriptions of the available adaptors.

get_adaptor_name()
Get the name of the adaptor that created this Scheduler.

classmethod get_adaptor_names()
Gives a list names of the available adaptors.

get_default_queue_name()
Get the name of the default queue.

get_file_system()
Retrieve the FileSystem used internally by this Scheduler.

get_job_status(*job=None*)
Get the status of a Job.

Parameters **job** (xenon.Job) – job

get_job_statuses(*jobs=None*)
Get the status of all specified jobs.

Parameters **jobs** (xenon.Job) – jobs

get_jobs(*queues=None*)
Get all job identifier of jobs currently in (one ore more) queues.

Parameters **queues** (*string*) – queues

get_location()
Get the location that this Scheduler is connected to.

get_properties()
Get the properties used to create this Scheduler.

get_queue_names()
Get the queue names supported by this Scheduler.

get_queue_status(*queue=None*)
Get the status of the queue.

Parameters **queue** (*string*) – queue

get_queue_statuses(*queues=None*)
Get the status of all queues.

Parameters **queues** (*string*) – queues

is_open()
Test if the connection of this Scheduler is open.

classmethod list_schedulers()
List the created schedulers Specific to grpc, not part of Xenon library

classmethod local_scheduler()
Get scheduler on local filesystem with default location, credential and no properties Not part of Scheduler class in Xenon library In Xenon library available as Scheduler.create("local")

submit_batch_job(*description=None*)
Submit a batch job.

Parameters **description** (xenon.JobDescription) – description

submit_interactive_job(*description*, *stdin_stream*)
Submit an interactive job The first response message in the response stream will contain the job identifier and empty stdout and stdout. Other response messages will also contain the job identifier and filled stdout and/or stderr.

wait_until_done(*job=None*, *timeout=None*)
Wait until a job is done or until a timeout expires.

Parameters

- **job** (xenon.Job) – job
- **timeout** (*uint64*) – timeout

wait_until_running(*job=None*, *timeout=None*)
Wait until a job starts running, or until a timeout expires.

Parameters

- **job** (xenon.Job) – job
- **timeout** (*uint64*) – timeout

4.3.1 Message classes

class xenon.**Job**(*id_*)
Job.

Variables **id** (*string*) – the Xenon job identifyer.

class xenon.**JobDescription**(***kwargs*)
This class describes a job to a Scheduler instance.

Variables

- **executable** (*string*) – executable
- **arguments** (*string*) – arguments
- **working_directory** (*string*) – working_directory
- **environment** (*map<string, string>*) – environment
- **queue_name** (*string*) – queue_name
- **max_runtime** (*uint32*) – max_runtime
- **node_count** (*uint32*) – node_count
- **processes_per_node** (*uint32*) – processes_per_node
- **start_single_process** (*bool*) – start_single_process
- **stderr** (*string*) – stderr
- **stdin** (*string*) – stdin
- **stdout** (*string*) – stdout
- **options** (*map<string, string>*) – options
- **name** (*string*) – name
- **max_memory** (*uint32*) – max_memory
- **scheduler_arguments** (*string*) – scheduler_arguments

class xenon.**JobStatus**(*service*, *wrapped*)
Status of a job.

Variables

- **job** (xenon.Job) – job
- **state** (*string*) – state
- **running** (*bool*) – running
- **done** (*bool*) – done
- **scheduler_specific_information** (*map<string, string>*) – scheduler_specific_information
- **exit_code** (*int32*) – exit_code

- **error_message** (*string*) – error_message
- ***error_type*** (xenon.JobStatus.ErrorType) – error_type
- **name** (*string*) – name

class ErrorType
An enumeration.

CANCELLED = 2

IO = 5

NONE = 0

NOT_CONNECTED = 3

NOT_FOUND = 1

OTHER = 6

XENON = 4

error_type

class xenon.**QueueStatus**(*service*, *wrapped*)
Status of a queue.

Variables

- **name** (*string*) – name
- **scheduler_specific_information** (*map<string, string>*) – scheduler_specific_information
- **error_message** (*string*) – error_message
- ***error_type*** (xenon.QueueStatus.ErrorType) – error_type

class ErrorType
An enumeration.

IO = 4

NONE = 0

NOT_CONNECTED = 2

NOT_FOUND = 1

OTHER = 5

XENON = 3

error_type

4.4 Credentials

class xenon.**CertificateCredential**

class xenon.**PasswordCredential**

4.5 Exceptions

exception xenon.exceptions.**AttributeNotSupportedException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**CopyCancelledException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**DirectoryNotEmptyException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**FileSystemClosedException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**IncompleteJobDescriptionException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**InvalidCredentialException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**InvalidJobDescriptionException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**InvalidLocationException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**InvalidOptionsException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**InvalidPathException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**InvalidPropertyException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**InvalidResumeTargetException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**NoSuchCopyException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**NoSuchJobException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**NoSuchPathException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**NoSuchQueueException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**PathAlreadyExistsException**(*method*, *exc_code*, *exc_msg*)
Exception that is raised if `FileSystem.create_directory()` fails due to an existing path.

exception xenon.exceptions.**PropertyTypeException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**UnknownAdaptorException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**UnknownPropertyException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**UnknownRpcException**(*method*, *exc_code*, *exc_msg*)
Default exception if nothing is known.

exception xenon.exceptions.**UnsupportedJobDescriptionException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**UnsupportedOperationException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

exception xenon.exceptions.**XenonException**(*method*, *code*, *msg*)
Xenon base exception.

exception xenon.exceptions.**XenonRuntimeException**(*method*, *exc_code*, *exc_msg*)
TODO: add doc-string.

xenon.exceptions.**make_exception**(*method*, *e*)
Creates an exception for a given method, and RpcError.

The PyXenon module interfaces with Xenon-GRPC to get an interface to the Xenon 2.0 Java library. We kept this interface close to the original Java API. PyXenon 2.0 only works on Python 3.

CHAPTER 5

Installing

```
pip install pyxenon
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

x

Index

J

L

M

N

O

P

Q

R

S

U

W

X