B2_Python_SDK

Release 1.0.2

Backblaze

Oct 16, 2019

CONTENTS

b2sdk is a client library for easy access to all of the capabilities of B2 Cloud Storage.

B2 command-line tool is an example of how it can be used to provide command-line access to the B2 service, but there are many possible applications (including FUSE filesystems, storage backend drivers for backup applications etc).

CHAPTER

ONE

WHY USE B2SDK?

When building an application which uses B2 cloud, it is possible to implement an independent B2 API client, but using **b2sdk** allows for:

- reuse of code that is already written, with hundreds of unit tests
- use **Syncronizer**, a high-performance, parallel rsync-like utility
- developer-friendly library *api version policy* which guards your program against incompatible changes
- B2 integration checklist is passed automatically
- **raw_simulator** makes it easy to mock the B2 cloud for unit testing purposes
- progress of operations will be reported to an object of your choice
- exception hierarchy makes it easy to display informative messages to users
- interrupted transfers are automatically continued
- **b2sdk** has been developed for 3 years before it version 1.0.0 was released. It's stable and mature.

CHAPTER

TWO

DOCUMENTATION INDEX

2.1 Installation Guide

2.1.1 Installing as a dependency

b2sdk can simply be added to `requirements.txt` (or equivalent such as `setup.py`, `.pipfile` etc). In order to properly set a dependency, see *versioning chapter* for details.

Note: The stability of your application depends on correct *pinning of versions*.

2.1.2 Installing a development version

To install **b2sdk**, checkout the repository and run:

```
pip install b2sdk
```

in your python environment.

Note: If you see a message saying that the `six` library cannot be installed, which happens if you're installing with the system python on OS X El Capitan, try this:

```
pip install --ignore-installed b2sdk
```

2.1.3 Installing for contributors

You'll need to some Python packages installed. To get all the latest things:

```
pip install --upgrade --upgrade-strategy eager -r requirements.txt -r requirements-
↪test.txt -r requirements-setup.txt
```

There is a *Makefile* with a rule to run the unit tests using the currently active Python:

```
make setup
make test
```

will install the required packages, then run the unit tests.

2.2 Tutorial

2.2.1 AccountInfo

`AccountInfo` object holds information about access keys, tokens, upload urls, as well as a bucket id-name map.

It is the first object that you need to create to use **b2sdk**. Using `AccountInfo`, we'll be able to create a `B2Api` object to manage a B2 account.

In the tutorial we will use *b2sdk.v1.InMemoryAccountInfo*:

```
>>> from b2sdk.v1 import InMemoryAccountInfo
>>> info = InMemoryAccountInfo()  # store credentials, tokens and cache in memory
```

With the `info` object in hand, we can now proceed to create a `B2Api` object.

Note: *AccountInfo* section provides guidance for choosing the correct `AccountInfo` class for your application.

2.2.2 Account authorization

```
>>> application_key_id = '4a5b6c7d8e9f'
>>> application_key = '001b8e23c26ff6efb941e237deb182b9599a84bef7'
>>> b2_api.authorize_account("production", application_key_id, application_key)
```

Tip: Get credentials from B2 website

To find out more about account authorization, see *b2sdk.v1.B2Api.authorize_account()*

2.2.3 B2Api

B2Api allows for account-level operations on a B2 account.

Typical B2Api operations

authorize_account	Perform account authorization.
create_bucket	Create a bucket.
delete_bucket	Delete a chosen bucket.
list_buckets	Call `b2_list_buckets` and return a list of buckets.
get_bucket_by_name	Return the Bucket matching the given bucket_name.
create_key	Create a new *application key*.
list_keys	List application keys.
delete_key	Delete *application key*.
download_file_by_id	Download a file with the given ID.
list_parts	Generator that yields a *b2sdk.v1.Part* for each of the parts that have been uploaded.
cancel_large_file	Cancel a large file upload.

```
>>> b2_api = B2Api(info)
```

to find out more, see *b2sdk.v1.B2Api*.

The most practical operation on `B2Api` object is *b2sdk.v1.B2Api.get_bucket_by_name()*.

Bucket allows for operations such as listing a remote bucket or transferring files.

2.2.4 Bucket

Initializing a Bucket

Retrieve an existing Bucket

To get a `Bucket` object for an existing B2 Bucket:

```
>>> b2_api.get_bucket_by_name("example-mybucket-b2-1",)
Bucket<346501784642eb3e60980d10,example-mybucket-b2-1,allPublic>
```

Create a new Bucket

To create a bucket:

```
>>> bucket_name = 'example-mybucket-b2-1'
>>> bucket_type = 'allPublic'  # or 'allPrivate'

>>> b2_api.create_bucket(bucket_name, bucket_type)
Bucket<346501784642eb3e60980d10,example-mybucket-b2-1,allPublic>
```

You can optionally store bucket info, CORS rules and lifecycle rules with the bucket. See *b2sdk.v1.B2Api.create_bucket()* for more details.

Note: Bucket name must be unique in B2 (across all accounts!). Your application should be able to cope with a bucket name collision with another B2 user.

Typical Bucket operations

download_file_by_name	Download a file by name.
upload_local_file	Upload a file on local disk to a B2 file.
upload_bytes	Upload bytes in memory to a B2 file.
ls	Pretend that folders exist and yields the information about the files in a folder.
hide_file	Hide a file.
delete_file_version	Delete a file version.
get_download_authorization	Return an authorization token that is valid only for downloading files from the given bucket.
get_download_url	Get file download URL.
update	Update various bucket parameters.

Continued on next page

Table 2 – continued from previous page

set_type	Update bucket type.
set_info	Update bucket info.

To find out more, see *b2sdk.v1.Bucket*.

2.2.5 Summary

You now know how to use `AccountInfo`, `B2Api` and `Bucket` objects.

To see examples of some of the methods presented above, visit the *quick start guide* section.

2.3 Quick Start Guide

2.3.1 Prepare b2sdk

```
>>> from b2sdk.v1 import *
>>> info = InMemoryAccountInfo()
>>> b2_api = B2Api(info)
>>> application_key_id = '4a5b6c7d8e9f'
>>> application_key = '001b8e23c26ff6efb941e237deb182b9599a84bef7'
>>> b2_api.authorize_account("production", application_key_id, application_key)
```

Tip: Get credentials from B2 website

2.3.2 Synchronization

```
>>> from b2sdk.v1 import ScanPoliciesManager
>>> from b2sdk.v1 import parse_sync_folder
>>> from b2sdk.v1 import Synchronizer
>>> import time
>>> import sys

>>> source = '/home/user1/b2_example'
>>> destination = 'b2://example-mybucket-b2'

>>> source = parse_sync_folder(source, b2_api)
>>> destination = parse_sync_folder(destination, b2_api)

>>> policies_manager = ScanPoliciesManager(exclude_all_symlinks=True)

>>> synchronizer = Synchronizer(
        max_workers=10,
        policies_manager=policies_manager,
        dry_run=False,
        allow_empty_source=True,
    )

>>> no_progress = False
```

(continues on next page)

(continued from previous page)

```
>>> with SyncReport(sys.stdout, no_progress) as reporter:
        synchronizer.sync_folders(
            source_folder=source,
            dest_folder=destination,
            now_millis=int(round(time.time() * 1000)),
            reporter=reporter,
        )
upload some.pdf
upload som2.pdf
```

Tip: Sync is the preferred way of getting data into and out of B2 cloud, because it can achieve *highest performance* due to parallelization of scanning and data transfer operations.

To learn more about sync, see *Synchronizer*.

2.3.3 Bucket actions

List buckets

```
>>> b2_api.list_buckets()
[Bucket<346501784642eb3e60980d10,example-mybucket-b2-1,allPublic>]
>>> for b in b2_api.list_buckets():
        print('%s  %-10s  %s' % (b.id_, b.type_, b.name))
346501784642eb3e60980d10  allPublic   example-mybucket-b2-1
```

Create a bucket

```
>>> bucket_name = 'example-mybucket-b2-1'  # must be unique in B2 (across all␣
↪accounts!)
>>> bucket_type = 'allPublic'  # or 'allPrivate'

>>> b2_api.create_bucket(bucket_name, bucket_type)
Bucket<346501784642eb3e60980d10,example-mybucket-b2-1,allPublic>
```

You can optionally store bucket info, CORS rules and lifecycle rules with the bucket. See *b2sdk.v1.B2Api.create_bucket()*.

Delete a bucket

```
>>> bucket_name = 'example-mybucket-b2-to-delete'
>>> bucket = b2_api.get_bucket_by_name(bucket_name)
>>> b2_api.delete_bucket(bucket)
```

returns *None* if successful, raises an exception in case of error.

Update bucket info

```
>>> new_bucket_type = 'allPrivate'
>>> bucket_name = 'example-mybucket-b2'

>>> bucket = b2_api.get_bucket_by_name(bucket_name)
>>> bucket.update(bucket_type=new_bucket_type)
{'accountId': '451862be08d0',
 'bucketId': '5485a1682662eb3e60980d10',
 'bucketInfo': {},
 'bucketName': 'example-mybucket-b2',
 'bucketType': 'allPrivate',
 'corsRules': [],
 'lifecycleRules': [],
 'revision': 3}
```

For more information see *b2sdk.v1.Bucket.update()*.

2.3.4 File actions

Tip: Sync is the preferred way of getting files into and out of B2 cloud, because it can achieve *highest performance* due to parallelization of scanning and data transfer operations.

To learn more about sync, see Sync.

Use the functions described below only if you *really* need to transfer a single file.

Upload file

```
>>> local_file_path = '/home/user1/b2_example/new.pdf'
>>> b2_file_name = 'dummy_new.pdf'
>>> file_info = {'how': 'good-file'}

>>> bucket = b2_api.get_bucket_by_name(bucket_name)
>>> bucket.upload_local_file(
        local_file=local_file_path,
        file_name=b2_file_name,
        file_infos=file_info,
    )
<b2sdk.file_version.FileVersionInfo at 0x7fc8cd560550>
```

This will work regardless of the size of the file - `upload_local_file` automatically uses large file upload API when necessary.

For more information see *b2sdk.v1.Bucket.upload_local_file()*.

Download file

By id

```
>>> from b2sdk.v1 import DownloadDestLocalFile
>>> from b2sdk.v1 import DoNothingProgressListener

>>> local_file_path = '/home/user1/b2_example/new2.pdf'
>>> file_id = '4_z5485a1682662eb3e60980d10_f1195145f42952533_d20190403_m130258_c002_
↪v0001111_t0002'
>>> download_dest = DownloadDestLocalFile(local_file_path)
>>> progress_listener = DoNothingProgressListener()

>>> b2_api.download_file_by_id(file_id, download_dest, progress_listener)
{'fileId': '4_z5485a1682662eb3e60980d10_f1195145f42952533_d20190403_m130258_c002_
↪v0001111_t0002',
 'fileName': 'som2.pdf',
 'contentType': 'application/pdf',
 'contentLength': 1870579,
 'contentSha1': 'd821849a70922e87c2b0786c0be7266b89d87df0',
 'fileInfo': {'src_last_modified_millis': '1550988084299'}}

>>> print('File name:   ', download_dest.file_name)
File name:    som2.pdf
>>> print('File id:     ', download_dest.file_id)
File id:      4_z5485a1682662eb3e60980d10_f1195145f42952533_d20190403_m130258_c002_
↪v0001111_t0002
>>> print('File size:   ', download_dest.content_length)
File size:    1870579
>>> print('Content type:', download_dest.content_type)
Content type: application/pdf
>>> print('Content sha1:', download_dest.content_sha1)
Content sha1: d821849a70922e87c2b0786c0be7266b89d87df0
```

By name

```
>>> bucket = b2_api.get_bucket_by_name(bucket_name)
>>> b2_file_name = 'dummy_new.pdf'
>>> local_file_name = '/home/user1/b2_example/new3.pdf'
>>> download_dest = DownloadDestLocalFile(local_file_name)
>>> bucket.download_file_by_name(b2_file_name, download_dest)
{'fileId': '4_z5485a1682662eb3e60980d10_f113f963288e711a6_d20190404_m065910_c002_
↪v0001095_t0044',
 'fileName': 'dummy_new.pdf',
 'contentType': 'application/pdf',
 'contentLength': 1870579,
 'contentSha1': 'd821849a70922e87c2b0786c0be7266b89d87df0',
 'fileInfo': {'how': 'good-file'}}
```

List files

```
>>> bucket_name = 'example-mybucket-b2'
>>> bucket = b2_api.get_bucket_by_name(bucket_name)
>>> for file_info, folder_name in bucket.ls(show_versions=False):
>>>     print(file_info.file_name, file_info.upload_timestamp, folder_name)
f2.txt 1560927489000 None
som2.pdf 1554296578000 None
```

(continues on next page)

(continued from previous page)

```
some.pdf 1554296579000 None
test-folder/.bzEmpty 1561005295000 test-folder/

# Recursive
>>> bucket_name = 'example-mybucket-b2'
>>> bucket = b2_api.get_bucket_by_name(bucket_name)
>>> for file_info, folder_name in bucket.ls(show_versions=False, recursive=True):
>>>     print(file_info.file_name, file_info.upload_timestamp, folder_name)
f2.txt 1560927489000 None
som2.pdf 1554296578000 None
some.pdf 1554296579000 None
test-folder/.bzEmpty 1561005295000 test-folder/
test-folder/folder_file.txt 1561005349000 None
```

Note: The files are returned recursively and in order so all files in a folder are printed one after another. The folder_name is returned only for the first file in the folder.

```
# Within folder
>>> bucket_name = 'example-mybucket-b2'
>>> bucket = b2_api.get_bucket_by_name(bucket_name)
>>> for file_info, folder_name in bucket.ls(folder_to_list='test-folder', show_
↪versions=False):
>>>     print(file_info.file_name, file_info.upload_timestamp, folder_name)
test-folder/.bzEmpty 1561005295000 None
test-folder/folder_file.txt 1561005349000 None

# list file versions
>>> for file_info, folder_name in bucket.ls(show_versions=True):
>>>     print(file_info.file_name, file_info.upload_timestamp, folder_name)
f2.txt 1560927489000 None
f2.txt 1560849524000 None
som2.pdf 1554296578000 None
some.pdf 1554296579000 None
```

For more information see *b2sdk.v1.Bucket.ls()*.

Get file metadata

```
>>> file_id = '4_z5485a1682662eb3e60980d10_f113f963288e711a6_d20190404_m065910_c002_
↪v0001095_t0044'
>>> b2_api.get_file_info(file_id)
{'accountId': '451862be08d0',
 'action': 'upload',
 'bucketId': '5485a1682662eb3e60980d10',
 'contentLength': 1870579,
 'contentSha1': 'd821849a70922e87c2b0786c0be7266b89d87df0',
 'contentType': 'application/pdf',
 'fileId': '4_z5485a1682662eb3e60980d10_f113f963288e711a6_d20190404_m065910_c002_
↪v0001095_t0044',
 'fileInfo': {'how': 'good-file'},
 'fileName': 'dummy_new.pdf',
 'uploadTimestamp': 1554361150000}
```

Copy file

```
>>> file_id = '4_z5485a1682662eb3e60980d10_f118df9ba2c5131e8_d20190619_m065809_c002_
↪v0001126_t0040'
>>> bucket.copy_file(file_id, 'f2_copy.txt')
{'accountId': '451862be08d0',
 'action': 'copy',
 'bucketId': '5485a1682662eb3e60980d10',
 'contentLength': 124,
 'contentSha1': '737637702a0e41dda8b7be79c8db1d369c6eef4a',
 'contentType': 'text/plain',
 'fileId': '4_z5485a1682662eb3e60980d10_f1022e2320daf707f_d20190620_m122848_c002_
↪v0001123_t0020',
 'fileInfo': {'src_last_modified_millis': '1560848707000'},
 'fileName': 'f2_copy.txt',
 'uploadTimestamp': 1561033728000}
```

If you want to copy just the part of the file, then you can specify the bytes_range as a tuple.

```
>>> file_id = '4_z5485a1682662eb3e60980d10_f118df9ba2c5131e8_d20190619_m065809_c002_
↪v0001126_t0040'
>>> bucket.copy_file(file_id, 'f2_copy.txt', bytes_range=(8,15))
{'accountId': '451862be08d0',
 'action': 'copy',
 'bucketId': '5485a1682662eb3e60980d10',
 'contentLength': 8,
 'contentSha1': '274713be564aecaae8de362acb68658b576d0b40',
 'contentType': 'text/plain',
 'fileId': '4_z5485a1682662eb3e60980d10_f114b0c11b6b6e39e_d20190620_m122007_c002_
↪v0001123_t0004',
 'fileInfo': {'src_last_modified_millis': '1560848707000'},
 'fileName': 'f2_copy.txt',
 'uploadTimestamp': 1561033207000}
```

For more information see *b2sdk.v1.Bucket.copy_file()*.

Delete file

```
>>> file_id = '4_z5485a1682662eb3e60980d10_f113f963288e711a6_d20190404_m065910_c002_
↪v0001095_t0044'
>>> file_info = b2_api.delete_file_version(file_id, 'dummy_new.pdf')
>>> print(file_info)
{'file_id': '4_z5485a1682662eb3e60980d10_f113f963288e711a6_d20190404_m065910_c002_
↪v0001095_t0044',
 'file_name': 'dummy_new.pdf'}
```

Cancel large file uploads

```
>>> bucket = b2_api.get_bucket_by_name(bucket_name)
>>> for file_version in bucket.list_unfinished_large_files():
        bucket.cancel_large_file(file_version.file_id)
```

2.4 Glossary

account ID An identifier of the B2 account (not login). Looks like this: `4ba5845d7aaf`.

application key ID Since every *Account ID* can have multiple access keys associated with it, the keys need to be distinguished from each other. *application key ID* is an identifier of the access key. There are two types of keys: *master application key* and *non-master application key*.

application key The secret associated with an *application key ID*, used to authenticate with the server. Looks like this: `N2Zug0evLcHDlh_L0Z0AJhiGGdY` or `0a1bce5ea463a7e4b090ef5bd6bd82b851928ab2c6` or `K0014pbwo1zxcIVMnqSNTfWHReU/O3s`

master application key This is the first key you have access to, it is available on the B2 web application. This key has all capabilities, access to all buckets, and has no file prefix restrictions or expiration. The *application key ID* of the master key is the same as the *account ID*.

non-master application key A key which can have restricted capabilities, can only have access to a certain bucket or even to just part of it. See https://www.backblaze.com/b2/docs/application_keys.html to learn more. Looks like this: `0014aa9865d6f0000000000b0`

Bucket A container that holds files. You can think of buckets as the top-level folders in your B2 Cloud Storage account. There is no limit to the number of files in a bucket, but there is a limit of 100 buckets per account. See https://www.backblaze.com/b2/docs/buckets.html to learn more.

2.5 API

2.5.1 Interface versions

You might notice that the import structure provided in the documentation looks a little odd: `from b2sdk.v1 import ...`. The `.v1` part is used to keep the interface fluid without risk of breaking applications that use the old signatures. With new versions, **b2sdk** will provide functions with signatures matching the old ones, wrapping the new interface in place of the old one. What this means for a developer using **b2sdk**, is that it will just keep working.

It also means that **b2sdk** developers may change the interface in the future and will not need to maintain many branches and backport fixes to keep compatibility of for users of those old branches.

Some effort will be put into keeping the performance of the old interfaces, but in some situations old interfaces may have slightly degraded performance after an interface change. If performance target is absolutely critical to your application, please pin your dependencies strictly (using `b2sdk==X.Y.Z`).

2.5.2 Public interface

Public interface consists of **public** members of modules listed in *Public API* section. This should be used in 99% of use cases, it's enough to implement anything from a console tool to a FUSE filesystem.

Those modules will not **functionally** change in a backwards-incompatible way between non-major versions. See *interface versions* chapter for notes on performance.

Hint: If the current version of **b2sdk** is `4.5.6` and you only use the *public* interface, put this in your `requirements.txt` to be safe:

```
b2sdk>=4.5.6,<5.0.0
```

Note: `b2sdk.*._something` and `b2sdk.*.*._something`, while having a name beginning with an underscore, are **NOT** considered public interface.

2.5.3 Internal interface

Some rarely used features of B2 cloud are not implemented in **b2sdk**. Tracking usage of transactions and transferred data is a good example - if it is required, additional work would need to be put into a specialized internal interface layer to enable accounting and reporting.

b2sdk maintainers are *very supportive* in case someone wants to contribute an additional feature. Please consider adding it to the sdk, so that more people can use it. This way it will also receive our updates, unlike a private implementation which would not receive any updates unless you apply them manually (but that's a lot of work and we both know it's not going to happen). In practice, an implementation can be either shared or will quickly become outdated. The license of **b2sdk** is very permissive, but when considering whether to keep your patches private or public, please take into consideration the long-term cost of keeping up with a dynamic open-source project and/or the cost of missing the updates, especially those related to performance and reliability (as those are being actively developed in parallel to documentation).

Internal interface modules are listed in *API Internal* section.

Note: It is OK for you to use our internal interface (better that than copying our source files!), however, if you do, please pin your dependencies to **middle** version, as backwards-incompatible changes may be introduced in a non-major version.

Hint: If the current version of **b2sdk** is `4.5.6` and you are using the *internal* interface, put this in your requirements.txt:

```
b2sdk>=4.5.6,<4.6.0
```

Hint: Use *Quick Start Guide* to quickly jump to examples

2.6 API Reference

2.6.1 Interface types

b2sdk API is divided into two parts, *public* and *internal*. Please pay attention to which interface type you use.

Tip: *Pinning versions* properly ensures the stability of your application.

2.6.2 Public API

AccountInfo

AccountInfo stores basic information about the account, such as *Application Key ID* and *Application Key*, in order to let `b2sdk.v1.B2Api` perform authenticated requests.

There are two usable implementations provided by **b2sdk**:

- `b2sdk.v1.InMemoryAccountInfo` - a basic implementation with no persistence
- `b2sdk.v1.SqliteAccountInfo` - for console and GUI applications

They both provide the full *AccountInfo interface*.

Note: Backup applications and many server-side applications should *implement their own AccountInfo*, backed by the metadata/configuration database of the application.

AccountInfo implementations

InMemoryAccountInfo

AccountInfo with no persistence.

class b2sdk.v1.**InMemoryAccountInfo**

AccountInfo which keeps all data in memory.

Implements all methods of *AccountInfo interface*.

Hint: Usage of this class is appropriate for secure Web applications which do not wish to persist any user data.

Using this class for applications such as CLI, GUI or backup is discouraged, as `InMemoryAccountInfo` does not write down the authorization token persistently. That would be slow, as it would force the application to retrieve a new one on every command/click/backup start. Furthermore - an important property of *AccountInfo* is caching the `bucket_name:bucket_id` mapping; in case of `InMemoryAccountInfo` the cache will be flushed between executions of the program.

__init__()

The constructor takes no parameters.

SqliteAccountInfo

class b2sdk.v1.**SqliteAccountInfo**

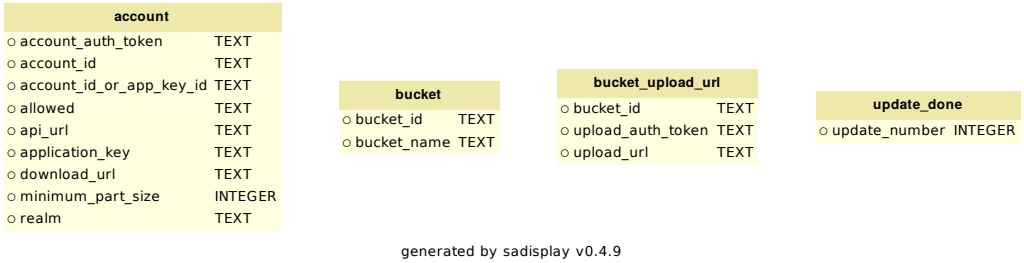
Store account information in an sqlite3 database which is used to manage concurrent access to the data.

The `update_done` table tracks the schema updates that have been completed.

Implements all methods of *AccountInfo interface*.

Uses a SQLite database for persistence and access synchronization between multiple processes. Not suitable for usage over NFS.

Underlying database has the following schema:



Hint: Usage of this class is appropriate for interactive applications installed on a user's machine (i.e.: CLI and GUI applications).

Usage of this class **might** be appropriate for non-interactive applications installed on the user's machine, such as backup applications. An alternative approach that should be considered is to store the *AccountInfo* data alongside the configuration of the rest of the application.

__init__ (*file_name=None*, *last_upgrade_to_run=None*)

If `file_name` argument is empty or `None`, path from `B2_ACCOUNT_INFO` environment variable is used. If that is not available, a default of `~/.b2_account_info` is used.

Parameters

- **file_name** (*str*) – The sqlite file to use; overrides the default.
- **last_upgrade_to_run** (*int*) – For testing only, override the auto-update on the db.

Implementing your own

When building a server-side application or a web service, you might want to implement your own *AccountInfo* class backed by a database. In such case, you should inherit from *b2sdk.v1.UrlPoolAccountInfo*, which has groundwork for url pool functionality). If you cannot use it, inherit directly from *b2sdk.v1.AbstractAccountInfo*.

```
>>> from b2sdk.v1 import UrlPoolAccountInfo
>>> class MyAccountInfo(UrlPoolAccountInfo):
        ...
```

b2sdk.v1.AbstractAccountInfo describes the interface, while *b2sdk.v1.UrlPoolAccountInfo* and `b2sdk.v1.UploadUrlPool` implement a part of the interface for in-memory upload token management.

AccountInfo interface

class b2sdk.v1.**AbstractAccountInfo**

Abstract class for a holder for all account-related information that needs to be kept between API calls and between invocations of the program.

This includes: account ID, application key ID, application key, auth tokens, API URL, download URL, and uploads URLs.

This class must be THREAD SAFE because it may be used by multiple threads running in the same Python process. It also needs to be safe against multiple processes running at the same time.

REALM_URLS = {'dev': 'http://api.backblazeb2.xyz:8180', 'production': 'https://api.b

DEFAULT_ALLOWED = {'bucketId': None, 'bucketName': None, 'capabilities': ['listKeys

classmethod all_capabilities()
Return a list of all possible capabilities.

Return type list

abstract clear()
Remove all stored information.

abstract refresh_entire_bucket_name_cache(*name_id_iterable*)
Remove all previous name-to-id mappings and stores new ones.

Parameters **name_id_iterable** (*iterable*) – an iterable of tuples of the form (name, id)

abstract remove_bucket_name(*bucket_name*)
Remove one entry from the bucket name cache.

Parameters **bucket_name** (*str*) – a bucket name

abstract save_bucket(*bucket*)
Remember the ID for the given bucket name.

Parameters **bucket** (b2sdk.v1.Bucket) – a Bucket object

abstract get_bucket_id_or_none_from_bucket_name(*bucket_name*)
Look up the bucket ID for the given bucket name.

Parameters **bucket_name** (*str*) – a bucket name

Return bucket ID or None

Return type str, None

abstract clear_bucket_upload_data(*bucket_id*)
Remove all upload URLs for the given bucket.

Parameters **bucket_id** (*str*) – a bucket ID

abstract get_account_id()
Return account ID or raises MissingAccountData exception.

Return type str

abstract get_application_key_id()
Return the application key ID used to authenticate.

Return type str

abstract get_account_auth_token()
Return account_auth_token or raises MissingAccountData exception.

Return type str

abstract get_api_url()
Return api_url or raises MissingAccountData exception.

Return type str

abstract get_application_key()
Return application_key or raises MissingAccountData exception.

Return type str

abstract get_download_url()
Return download_url or raises MissingAccountData exception.

Return type str

abstract get_realm()
Return realm or raises MissingAccountData exception.

Return type str

abstract get_minimum_part_size()
Return the minimum number of bytes in a part of a large file.

Returns number of bytes

Return type int

abstract get_allowed()
An 'allowed' dict, as returned by `b2_authorize_account`. Never `None`; for account info that was saved before 'allowed' existed, returns *DEFAULT_ALLOWED*.

Return type dict

set_auth_data(*account_id*, *auth_token*, *api_url*, *download_url*, *minimum_part_size*, *application_key*, *realm*, *allowed=None*, *application_key_id=None*)
Check permission correctness and stores the results of `b2_authorize_account`.

The allowed structure is the one returned by `b2_authorize_account` with an addition of a bucketName field. For keys with bucket restrictions, the name of the bucket is looked up and stored as well. The console_tool does everything by bucket name, so it's convenient to have the restricted bucket name handy.

Parameters

- **account_id** (*str*) – user account ID
- **auth_token** (*str*) – user authentication token
- **api_url** (*str*) – an API URL
- **download_url** (*str*) – path download URL
- **minimum_part_size** (*int*) – minimum size of the file part
- **application_key** (*str*) – application key
- **realm** (*str*) – a realm to authorize account in
- **allowed** (*dict*) – the structure to use for old account info that was saved without 'allowed'
- **application_key_id** (*str*) – application key ID

Changed in version 0.1.5: *account_id_or_app_key_id* renamed to *application_key_id*

classmethod allowed_is_valid(*allowed*)
Make sure that all of the required fields are present, and that bucketId is set if bucketName is.

If the bucketId is for a bucket that no longer exists, or the capabilities do not allow for listBuckets, then we will not have a bucketName.

Parameters **allowed** (*dict*) – the structure to use for old account info that was saved without 'allowed'

Return type bool

abstract _set_auth_data(*account_id*, *auth_token*, *api_url*, *download_url*, *minimum_part_size*, *application_key*, *realm*, *allowed*, *application_key_id*)
Actually store the auth data. Can assume that 'allowed' is present and valid.

All of the information returned by `b2_authorize_account` is saved, because all of it is needed at some point.

abstract take_bucket_upload_url(*bucket_id*)
Return a pair (upload_url, upload_auth_token) that has been removed from the pool for this bucket, or (None, None) if there are no more left.

Parameters **bucket_id** (*str*) – a bucket ID

Return type tuple

abstract put_bucket_upload_url(*bucket_id*, *upload_url*, *upload_auth_token*)
Add an (upload_url, upload_auth_token) pair to the pool available for the bucket.

Parameters

- **bucket_id** (*str*) – a bucket ID
- **upload_url** (*str*) – an upload URL
- **upload_auth_token** (*str*) – an upload authentication token

Return type tuple

abstract put_large_file_upload_url(*file_id*, *upload_url*, *upload_auth_token*)
Put a large file upload URL into a pool.

Parameters

- **file_id** (*str*) – a file ID
- **upload_url** (*str*) – an upload URL
- **upload_auth_token** (*str*) – an upload authentication token

abstract take_large_file_upload_url(*file_id*)
Take the chosen large file upload URL from the pool.

Parameters **file_id** (*str*) – a file ID

abstract clear_large_file_upload_urls(*file_id*)
Clear the pool of URLs for a given file ID.

Parameters **file_id** (*str*) – a file ID

AccountInfo helper classes

class b2sdk.v1.**UrlPoolAccountInfo**
Implement part of *AbstractAccountInfo* for upload URL pool management with a simple, key-value storage, such as `b2sdk.v1.UploadUrlPool`.

> **Caution:** This class is not part of the public interface. To find out how to safely use it, read *this*.

BUCKET_UPLOAD_POOL_CLASS
A url pool class to use for small files.

alias of `UploadUrlPool`

LARGE_FILE_UPLOAD_POOL_CLASS
A url pool class to use for large files.

alias of `UploadUrlPool`

class b2sdk.account_info.upload_url_pool.**UploadUrlPool**
For each key (either a bucket id or large file id), hold a pool of (url, auth_token) pairs.

> **Caution:** This class is not part of the public interface. To find out how to safely use it, read *this*.

put(*key*, *url*, *auth_token*)
Add the url and auth token to the pool for the given key.

Parameters

- **key** (*str*) – bucket ID or large file ID
- **url** (*str*) – bucket or file URL
- **auth_token** (*str*) – authentication token

take(*key*)
Return a (url, auth_token) if one is available, or (None, None) if not.

Parameters **key** (*str*) – bucket ID or large file ID

Return type tuple

clear_for_key(*key*)
Remove an item from the pool by key.

Parameters **key** (*str*) – bucket ID or large file ID

B2 Api client

class b2sdk.v1.**B2Api**
Provide file-level access to B2 services.

While `b2sdk.v1.B2RawApi` provides direct access to the B2 web APIs, this class handles several things that simplify the task of uploading and downloading files:

- re-acquires authorization tokens when they expire
- retrying uploads when an upload URL is busy
- breaking large files into parts
- emulating a directory structure (B2 buckets are flat)

Adds an object-oriented layer on top of the raw API, so that buckets and files returned are Python objects with accessor methods.

The class also keeps a cache of information needed to access the service, such as auth tokens and upload URLs.

BUCKET_FACTORY_CLASS
alias of `b2sdk.bucket.BucketFactory`

BUCKET_CLASS
alias of `b2sdk.bucket.Bucket`

__init__(*account_info=None*, *cache=None*, *raw_api=None*, *max_upload_workers=10*)
Initialize the API using the given account info.

Parameters

- **account_info** – an instance of `UrlPoolAccountInfo`, or any custom class derived from `AbstractAccountInfo`
- **cache** – an instance of the one of the following classes: *DummyCache*, *InMemoryCache*, *AuthInfoCache*, or any custom class derived from *AbstractCache*
- **raw_api** – an instance of one of the following classes: *B2RawApi*, `RawSimulator`, or any custom class derived from *AbstractRawApi*
- **max_upload_workers** (*int*) – a number of upload threads, default is 10

set_thread_pool_size(*max_workers*)
Set the size of the thread pool to use for uploads and downloads.

Must be called before any work starts, or the thread pool will get the default size of 1.

Parameters **max_workers** (*int*) – maximum allowed number of workers in a pool

get_thread_pool()
Return the thread pool executor to use for uploads and downloads.

authorize_automatically()
Perform automatic account authorization, retrieving all account data from account info object passed during initialization.

authorize_account(*realm*, *application_key_id*, *application_key*)
Perform account authorization.

Parameters

- **realm** (*str*) – a realm to authorize account in (usually just "production")
- **application_key_id** (*str*) – *application key ID*
- **application_key** (*str*) – user's *application key*

get_account_id()
Return the account ID.

Return type str

create_bucket(*name*, *bucket_type*, *bucket_info=None*, *cors_rules=None*, *lifecycle_rules=None*)
Create a bucket.

Parameters

- **name** (*str*) – bucket name
- **bucket_type** (*str*) – a bucket type, could be one of the following values: `"allPublic"`, `"allPrivate"`
- **bucket_info** (*dict*) – additional bucket info to store with the bucket
- **cors_rules** (*dict*) – bucket CORS rules to store with the bucket
- **lifecycle_rules** (*dict*) – bucket lifecycle rules to store with the bucket

Returns a Bucket object

Return type *b2sdk.v1.Bucket*

download_file_by_id(*file_id*, *download_dest*, *progress_listener=None*, *range_=None*)
Download a file with the given ID.

Parameters

- **file_id** (*str*) – a file ID
- **download_dest** (*str*) – a local file path
- **progress_listener** – an instance of the one of the following classes: PartProgressReporter, *TqdmProgressListener*, *SimpleProgressListener*, *DoNothingProgressListener*, *ProgressListenerForTest*, SyncFileReporter, or any sub class of *AbstractProgressListener*
- **range** (*list*) – a list of two integers, the first one is a start position, and the second one is the end position in the file

Returns context manager that returns an object that supports iter_content()

get_bucket_by_id(*bucket_id*)
Return a bucket object with a given ID. Unlike get_bucket_by_name, this method does not need to make any API calls.

Parameters **bucket_id** (*str*) – a bucket ID

Returns a Bucket object

Return type *b2sdk.v1.Bucket*

get_bucket_by_name(*bucket_name*)
Return the Bucket matching the given bucket_name.

Parameters **bucket_name** (*str*) – the name of the bucket to return

Returns a Bucket object

Return type *b2sdk.v1.Bucket*

Raises ***b2sdk.v1.exception.NonExistentBucket*** – if the bucket does not exist in the account

delete_bucket(*bucket*)
Delete a chosen bucket.

Parameters **bucket** (b2sdk.v1.Bucket) – a *Bucket* to delete

Return type None

list_buckets(*bucket_name=None*)
Call b2_list_buckets and return a list of buckets.

When no bucket name is specified, returns *all* of the buckets in the account. When a bucket name is given, returns just that bucket. When authorized with an *application key* restricted to one *bucket*, you must specify the bucket name, or the request will be unauthorized.

Parameters **bucket_name** (*str*) – the name of the one bucket to return

Return type list[*b2sdk.v1.Bucket*]

list_parts(*file_id*, *start_part_number=None*, *batch_size=None*)
Generator that yields a *b2sdk.v1.Part* for each of the parts that have been uploaded.

Parameters

- **file_id** (*str*) – the ID of the large file that is not finished

- **start_part_number** (*int*) – the first part number to return; defaults to the first part
- **batch_size** (*int*) – the number of parts to fetch at a time from the server

Return type generator

cancel_large_file(*file_id*)
Cancel a large file upload.

Parameters **file_id** (*str*) – a file ID

Return type None

delete_file_version(*file_id*, *file_name*)
Permanently and irrevocably delete one version of a file.

Parameters

- **file_id** (*str*) – a file ID
- **file_name** (*str*) – a file name

Return type *FileIdAndName*

get_download_url_for_fileid(*file_id*)
Return a URL to download the given file by ID.

Parameters **file_id** (*str*) – a file ID

get_download_url_for_file_name(*bucket_name*, *file_name*)
Return a URL to download the given file by name.

Parameters

- **bucket_name** (*str*) – a bucket name
- **file_name** (*str*) – a file name

create_key(*capabilities*, *key_name*, *valid_duration_seconds=None*, *bucket_id=None*, *name_prefix=None*)
Create a new *application key*.

Parameters

- **capabilities** (*list*) – a list of capabilities
- **key_name** (*str*) – a name of a key
- **valid_duration_seconds** (*int*, *None*) – key auto-expire time after it is created, in seconds, or None to not expire
- **bucket_id** (*str*, *None*) – a bucket ID to restrict the key to, or None to not restrict
- **name_prefix** (*str*, *None*) – a remote filename prefix to restrict the key to or None to not restrict

delete_key(*application_key_id*)
Delete *application key*.

Parameters **application_key_id** (*str*) – an *application key ID*

list_keys(*start_application_key_id=None*)
List application keys.

Parameters **start_application_key_id** (*str*, *None*) – an *application key ID* to start from or None to start from the beginning

get_file_info(*file_id*)
Legacy interface which just returns whatever remote API returns.

check_bucket_restrictions(*bucket_name*)
Check to see if the allowed field from authorize-account has a bucket restriction.

If it does, checks if the bucket_name for a given api call matches that. If not, it raises a *b2sdk.v1.exception.RestrictedBucket* error.

Parameters **bucket_name** (*str*) – a bucket name

Raises ***b2sdk.v1.exception.RestrictedBucket*** – if the account is not allowed to use this bucket

Exceptions

exception b2sdk.v1.exception.**AccountInfoError**(**args*, ***kwargs*)
Base class for all account info errors.

exception b2sdk.v1.exception.**CorruptAccountInfo**(*file_name*)
Raised when an account info file is corrupted.

exception b2sdk.v1.exception.**MissingAccountData**(*key*)
Raised when there is no account info data available.

exception b2sdk.v1.exception.**AlreadyFailed**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**B2ConnectionError**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**B2Error**(**args*, ***kwargs*)

property prefix
Nice, auto-generated error message prefix.

```
>>> B2SimpleError().prefix
'Simple error'
>>> AlreadyFailed().prefix
'Already failed'
```

should_retry_http()
Return true if this is an error that can cause an HTTP call to be retried.

should_retry_upload()
Return true if this is an error that should tell the upload code to get a new upload URL and try the upload again.

exception b2sdk.v1.exception.**B2HttpCallbackException**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**B2HttpCallbackPostRequestException**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**B2HttpCallbackPreRequestException**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**B2RequestTimeout**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**B2SimpleError**(**args*, ***kwargs*)
A B2Error with a message prefix.

exception b2sdk.v1.exception.**BadDateFormat**(**args*, ***kwargs*)

prefix = 'Date from server'

exception b2sdk.v1.exception.**BadFileInfo**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**BadJson**(**args*, ***kwargs*)

prefix = 'Bad request'

exception b2sdk.v1.exception.**BadUploadUrl**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**BrokenPipe**(**args*, ***kwargs*)

should_retry_upload()
Return true if this is an error that should tell the upload code to get a new upload URL and try the upload again.

exception b2sdk.v1.exception.**BucketNotAllowed**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**CapabilityNotAllowed**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**ChecksumMismatch**(*checksum_type*, *expected*, *actual*)

exception b2sdk.v1.exception.**ClockSkew**(*clock_skew_seconds*)
The clock on the server differs from the local clock by too much.

exception b2sdk.v1.exception.**CommandError**(*message*)
b2 command error (user caused). Accepts exactly one argument: message.

We expect users of shell scripts will parse our `__str__` output.

exception b2sdk.v1.exception.**Conflict**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**ConnectionReset**(**args*, ***kwargs*)

should_retry_upload()
Return true if this is an error that should tell the upload code to get a new upload URL and try the upload again.

exception b2sdk.v1.exception.**DestFileNewer**(*dest_file*, *source_file*, *dest_prefix*, *source_prefix*)

should_retry_http()
Return true if this is an error that can cause an HTTP call to be retried.

exception b2sdk.v1.exception.**DuplicateBucketName**(**args*, ***kwargs*)

prefix = 'Bucket name is already in use'

exception b2sdk.v1.exception.**FileAlreadyHidden**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**FileNameNotAllowed**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**FileNotPresent**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**InvalidAuthToken**(*message*, *code*)
Specific type of Unauthorized that means the auth token is invalid. This is not the case where the auth token is valid, but does not allow access.

exception b2sdk.v1.exception.**InvalidMetadataDirective**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**InvalidRange**(*content_length*, *range_*)

exception b2sdk.v1.exception.**InvalidUploadSource**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**MaxFileSizeExceeded**(*size*, *max_allowed_size*)

exception b2sdk.v1.exception.**MaxRetriesExceeded**(*limit*, *exception_info_list*)

exception b2sdk.v1.exception.**MissingPart**(**args*, ***kwargs*)

prefix = 'Part number has not been uploaded'

exception b2sdk.v1.exception.**NonExistentBucket**(**args*, ***kwargs*)

prefix = 'No such bucket'

exception b2sdk.v1.exception.**NotAllowedByAppKeyError**(**args*, ***kwargs*)
Base class for errors caused by restrictions on an application key.

exception b2sdk.v1.exception.**PartSha1Mismatch**(*key*)

exception b2sdk.v1.exception.**RestrictedBucket**(*bucket_name*)

exception b2sdk.v1.exception.**ServiceError**(**args*, ***kwargs*)
Used for HTTP status codes 500 through 599.

exception b2sdk.v1.exception.**StorageCapExceeded**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**TooManyRequests**(*retry_after_seconds=None*)

should_retry_http()
Return true if this is an error that can cause an HTTP call to be retried.

class b2sdk.v1.exception.**TransientErrorMixin**

should_retry_http()

should_retry_upload()

exception b2sdk.v1.exception.**TruncatedOutput**(*bytes_read*, *file_size*)

exception b2sdk.v1.exception.**Unauthorized**(*message*, *code*)

should_retry_upload()
Return true if this is an error that should tell the upload code to get a new upload URL and try the upload again.

exception b2sdk.v1.exception.**UnexpectedCloudBehaviour**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**UnknownError**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**UnknownHost**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**UnrecognizedBucketType**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**UnsatisfiableRange**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**UnusableFileName**(**args*, ***kwargs*)
Raise when a filename doesn't meet the rules.

Could possibly use InvalidUploadSource, but this is intended for the filename on the server, which could differ. https://www.backblaze.com/b2/docs/files.html.

b2sdk.v1.exception.**interpret_b2_error**(*status*, *code*, *message*, *response_headers*, *post_params=None*)

exception b2sdk.v1.exception.**EnvironmentEncodingError**(*filename*, *encoding*)
Raised when a file name can not be decoded with system encoding.

exception b2sdk.v1.exception.**IncompleteSync**(**args*, ***kwargs*)

exception b2sdk.v1.exception.**InvalidArgument**(*parameter_name*, *message*)
Raised when one or more arguments are invalid

B2 Bucket

class b2sdk.v1.**Bucket**
Provide access to a bucket in B2: listing files, uploading and downloading.

DEFAULT_CONTENT_TYPE = 'b2/x-auto'

MAX_UPLOAD_ATTEMPTS = 5

MAX_LARGE_FILE_SIZE = 10000000000000

__init__(*api*, *id_*, *name=None*, *type_=None*, *bucket_info=None*, *cors_rules=None*, *lifecycle_rules=None*, *revision=None*, *bucket_dict=None*)

Parameters

- **api** (b2sdk.v1.B2Api) – an API object
- **id** (*str*) – a bucket id
- **name** (*str*) – a bucket name
- **type** (*str*) – a bucket type
- **bucket_info** (*dict*) – an info to store with a bucket
- **cors_rules** (*dict*) – CORS rules to store with a bucket
- **lifecycle_rules** (*dict*) – lifecycle rules to store with a bucket
- **revision** (*int*) – a bucket revision number
- **bucket_dict** (*dict*) – a dictionary which contains bucket parameters

get_id()
Return bucket ID.

Return type str

set_info(*new_bucket_info*, *if_revision_is=None*)
Update bucket info.

Parameters

- **new_bucket_info** (*dict*) – new bucket info dictionary
- **if_revision_is** (*int*) – revision number, update the info **only if** *revision* equals to *if_revision_is*

set_type(*bucket_type*)
Update bucket type.

Parameters **bucket_type** (*str*) – a bucket type (“allPublic” or “allPrivate”)

update(*bucket_type=None*, *bucket_info=None*, *cors_rules=None*, *lifecycle_rules=None*, *if_revision_is=None*)
Update various bucket parameters.

Parameters

- **bucket_type** (*str*) – a bucket type
- **bucket_info** (*dict*) – an info to store with a bucket
- **cors_rules** (*dict*) – CORS rules to store with a bucket
- **lifecycle_rules** (*dict*) – lifecycle rules to store with a bucket
- **if_revision_is** (*int*) – revision number, update the info **only if** *revision* equals to *if_revision_is*

cancel_large_file(*file_id*)
Cancel a large file transfer.

Parameters **file_id** (*str*) – a file ID

download_file_by_id(*file_id*, *download_dest*, *progress_listener=None*, *range_=None*)
Download a file by ID.

Note: download_file_by_id actually belongs in *b2sdk.v1.B2Api*, not in *b2sdk.v1.Bucket*; we just provide a convenient redirect here

---

Parameters

- **file_id** (*str*) – a file ID
- **download_dest** (*str*) – a local file path
- **None progress_listener** (b2sdk.v1.AbstractProgressListener,) – a progress listener object to use, or None to not report progress
- **int] range** (*tuple[int,*) – two integer values, start and end offsets

download_file_by_name(*file_name*, *download_dest*, *progress_listener=None*, *range_=None*)
Download a file by name.

See also:

Synchronizer, a *high-performance* utility that synchronizes a local folder with a Bucket.

Parameters

- **file_id** (*str*) – a file ID
- **download_dest** (*str*) – a local file path
- **None progress_listener** (b2sdk.v1.AbstractProgressListener,) – a progress listener object to use, or None to not track progress
- **int] range** (*tuple[int,*) – two integer values, start and end offsets

get_download_authorization(*file_name_prefix*, *valid_duration_in_seconds*)
Return an authorization token that is valid only for downloading files from the given bucket.

Parameters

- **file_name_prefix** (*str*) – a file name prefix, only files that match it could be downloaded
- **valid_duration_in_seconds** (*int*) – a token is valid only during this amount of seconds

list_parts (*file_id*, *start_part_number=None*, *batch_size=None*)
Get a list of all parts that have been uploaded for a given file.

Parameters

- **file_id** (*str*) – a file ID
- **start_part_number** (*int*) – the first part number to return. defaults to the first part.
- **batch_size** (*int*) – the number of parts to fetch at a time from the server

ls (*folder_to_list=''*, *show_versions=False*, *recursive=False*, *fetch_count=None*)
Pretend that folders exist and yields the information about the files in a folder.

B2 has a flat namespace for the files in a bucket, but there is a convention of using "/" as if there were folders. This method searches through the flat namespace to find the files and "folders" that live within a given folder.

When the *recursive* flag is set, lists all of the files in the given folder, and all of its sub-folders.

Parameters

- **folder_to_list** (*str*) – the name of the folder to list; must not start with "/". Empty string means top-level folder
- **show_versions** (*bool*) – when `True` returns info about all versions of a file, when `False`, just returns info about the most recent versions
- **recursive** (*bool*) – if `True`, list folders recursively
- **fetch_count** (*int,None*) – how many entries to return or `None` to use the default. Acceptable values: 1 - 1000

Return type generator[tuple[*b2sdk.v1.FileVersionInfo*, str]]

Returns generator of (file_version_info, folder_name) tuples

Note: In case of *recursive=True*, folder_name is returned only for first file in the folder.

---

list_unfinished_large_files (*start_file_id=None*, *batch_size=None*)
A generator that yields an *b2sdk.v1.UnfinishedLargeFile* for each unfinished large file in the bucket, starting at the given file.

Parameters

- **start_file_id** (*str,None*) – a file ID to start from or None to start from the beginning
- **batch_size** (*int,None*) – max file count

Return type generator[*b2sdk.v1.UnfinishedLargeFile*]

start_large_file (*file_name*, *content_type=None*, *file_info=None*)
Start a large file transfer.

Parameters

- **file_name** (*str*) – a file name
- **content_type** (*str,None*) – the MIME type, or `None` to accept the default based on file extension of the B2 file name
- **file_infos** (*dict,None*) – a file info to store with the file or `None` to not store anything

upload_bytes(*data_bytes*, *file_name*, *content_type=None*, *file_infos=None*, *progress_listener=None*)

Upload bytes in memory to a B2 file.

Parameters

- **data_bytes** (*bytes*) – a byte array to upload
- **file_name** (*str*) – a file name to upload bytes to
- **content_type** (*str*, *None*) – the MIME type, or None to accept the default based on file extension of the B2 file name
- **file_infos** (*dict*, *None*) – a file info to store with the file or None to not store anything
- **progress_listener** (b2sdk.v1.AbstractProgressListener, *None*) – a progress listener object to use, or None to not track progress

upload_local_file(*local_file*, *file_name*, *content_type=None*, *file_infos=None*, *sha1_sum=None*, *min_part_size=None*, *progress_listener=None*)

Upload a file on local disk to a B2 file.

See also:

Synchronizer, a *high-performance* utility that synchronizes a local folder with a *Bucket*.

Parameters

- **local_file** (*str*) – a path to a file on local disk
- **file_name** (*str*) – a file name of the new B2 file
- **content_type** (*str*, *None*) – the MIME type, or None to accept the default based on file extension of the B2 file name
- **file_infos** (*dict*, *None*) – a file info to store with the file or None to not store anything
- **sha1_sum** (*str*, *None*) – file SHA1 hash or None to compute it automatically
- **min_part_size** (*int*) – a minimum size of a part
- **progress_listener** (b2sdk.v1.AbstractProgressListener, *None*) – a progress listener object to use, or None to not report progress

upload(*upload_source*, *file_name*, *content_type=None*, *file_info=None*, *min_part_size=None*, *progress_listener=None*)

Upload a file to B2, retrying as needed.

The source of the upload is an UploadSource object that can be used to open (and re-open) the file. The result of opening should be a binary file whose read() method returns bytes.

Parameters

- **upload_source** (*b2sdk.v1.UploadSource*) – an object that opens the source of the upload
- **file_name** (*str*) – the file name of the new B2 file
- **content_type** (*str*, *None*) – the MIME type, or None to accept the default based on file extension of the B2 file name
- **file_infos** (*dict*, *None*) – a file info to store with the file or None to not store anything

- **min_part_size** (*int*, *None*) – the smallest part size to use or None to determine automatically
- **progress_listener** (b2sdk.v1.AbstractProgressListener, *None*) – a progress listener object to use, or None to not report progress

The function *opener* should return a file-like object, and it must be possible to call it more than once in case the upload is retried.

get_download_url(*filename*)
Get file download URL.

Parameters **filename** (*str*) – a file name

Return type str

hide_file(*file_name*)
Hide a file.

Parameters **file_name** (*str*) – a file name

Return type *b2sdk.v1.FileVersionInfo*

copy_file(*file_id*, *new_file_name*, *bytes_range=None*, *metadata_directive=None*, *content_type=None*, *file_info=None*)
Creates a new file in this bucket by (server-side) copying from an existing file.

Parameters

- **file_id** (*str*) – file ID of existing file
- **new_file_name** (*str*) – file name of the new file
- **bytes_range** (*tuple[int, int]*, *None*) – start and end offsets (**inclusive!**), default is the entire file
- **metadata_directive** (*b2sdk.v1.MetadataDirectiveMode*, *None*) – default is b2sdk.v1.MetadataDirectiveMode.COPY
- **content_type** (*str*, *None*) – content_type for the new file if metadata_directive is set to b2sdk.v1.MetadataDirectiveMode.REPLACE, default will copy the content_type of old file
- **file_info** (*dict*, *None*) – file_info for the new file if metadata_directive is set to b2sdk.v1.MetadataDirectiveMode.REPLACE, default will copy the file_info of old file

delete_file_version(*file_id*, *file_name*)
Delete a file version.

Parameters

- **file_id** (*str*) – a file ID
- **file_name** (*str*) – a file name

as_dict()
Return bucket representation as a dictionary.

Return type dict

Data classes

class b2sdk.v1.**FileVersionInfo**(*id_*, *file_name*, *size*, *content_type*, *content_sha1*, *file_info*, *upload_timestamp*, *action*)

A structure which represents a version of a file (in B2 cloud).

Variables

- **id_** (*str*) – fileId
- **file_name** (*str*) – full file name (with path)
- **size** (*int or None*) – size in bytes, can be None (unknown)
- **content_type** (*str*) – RFC 822 content type, for example "application/octet-stream"
- **content_sha1** (*str or None*) – sha1 checksum of the entire file, can be None (unknown) if it is a large file uploaded by a client which did not provide it
- **file_info** (*dict*) – file info dict
- **upload_timestamp** (*int or None*) – in milliseconds since EPOCH (1970-01-01 00:00:00). Can be None (unknown).
- **action** (*str*) – "upload", "hide" or "delete"

class b2sdk.v1.**FileIdAndName**(*file_id*, *file_name*)

A structure which represents a B2 cloud file with just *file_name* and *fileId* attributes.

Used to return data from calls to *b2sdk.v1.Bucket.delete_file_version()*.

Variables

- **file_id** (*str*) – fileId
- **file_name** (*str*) – full file name (with path)

class b2sdk.v1.**UnfinishedLargeFile**

A structure which represents a version of a file (in B2 cloud).

Variables

- **file_id** (*str*) – fileId
- **file_name** (*str*) – full file name (with path)
- **account_id** (*str*) – account ID
- **bucket_id** (*str*) – bucket ID
- **content_type** (*str*) – **RFC 822** content type, for example "application/octet-stream"
- **file_info** (*dict*) – file info dict

class b2sdk.v1.**Part**(*file_id*, *part_number*, *content_length*, *content_sha1*)

A structure which represents a *part* of a large file upload.

Variables

- **file_id** (*str*) – fileId
- **part_number** (*int*) – part number, starting with 1
- **content_length** (*str*) – content length, in bytes
- **content_sha1** (*str*) – checksum

Progress reporters

Note: Concrete classes described in this chapter implement methods defined in `AbstractProgressListener`

class b2sdk.v1.**AbstractProgressListener**
Interface expected by B2Api upload and download methods to report on progress.

This interface just accepts the number of bytes transferred so far. Subclasses will need to know the total size if they want to report a percent done.

abstract set_total_bytes(*total_byte_count*)
Always called before __enter__ to set the expected total number of bytes.

May be called more than once if an upload is retried.

Parameters **total_byte_count** (*int*) – expected total number of bytes

abstract bytes_completed(*byte_count*)
Report the given number of bytes that have been transferred so far. This is not a delta, it is the total number of bytes transferred so far.

Parameters **byte_count** (*int*) – number of bytes have been transferred

abstract close()
Must be called when you're done with the listener. In well-structured code, should be called only once.

class b2sdk.v1.**TqdmProgressListener**(*description*, **args*, ***kwargs*)
Progress listener based on tqdm library.

class b2sdk.v1.**SimpleProgressListener**(*description*, **args*, ***kwargs*)
Just a simple progress listener which prints info on a console.

class b2sdk.v1.**DoNothingProgressListener**
This listener gives no output whatsoever.

class b2sdk.v1.**ProgressListenerForTest**(**args*, ***kwargs*)
Capture all of the calls so they can be checked.

Synchronizer

Synchronizer is a powerful utility with functionality of a basic backup application. It is able to copy entire folders into the cloud and back to a local drive, providing retention policies and many other options.

The **high performance** of sync is credited to parallelization of:

- listing local directory contents
- listing bucket contents
- uploads
- downloads

Synchronizer spawns threads to perform the operations listed above in parallel to shorten the backup window to a minimum.

Sync Options

Following are the important optional arguments that can be provided while initializing *Synchronizer* class.

- `compare_version_mode`: When comparing the source and destination files for finding whether to replace them or not, *compare_version_mode* can be passed to specify the mode of comparision. For possible values see `b2sdk.v1.CompareVersionMode`. Default value is `b2sdk.v1.CompareVersionMode.MODTIME`
- `compare_threshold`: It's the minimum size (in bytes)/modification time (in seconds) difference between source and destination files before we assume that it is new and replace.
- `newer_file_mode`: To identify whether to skip or replace if source is older. For possible values see `b2sdk.v1.NewerFileSyncMode`. If you don't specify this the sync will raise *b2sdk.v1.exception.DestFileNewer* in case any of the source file is older than destination.
- `keep_days_or_delete`: specify policy to keep or delete older files. For possible values see `b2sdk.v1.KeepOrDeleteMode`. Default is *DO_NOTHING*.
- `keep_days`: if *keep_days_or_delete* is `b2sdk.v1.CompareVersionMode.KEEP_BEFORE_DELETE` then this specify for how many days should we keep.

```
>>> from b2sdk.v1 import ScanPoliciesManager
>>> from b2sdk.v1 import parse_sync_folder
>>> from b2sdk.v1 import Synchronizer
>>> from b2sdk.v1 import KeepOrDeleteMode, CompareVersionMode, NewerFileSyncMode
>>> import time
>>> import sys

>>> source = '/home/user1/b2_example'
>>> destination = 'b2://example-mybucket-b2'

>>> source = parse_sync_folder(source, b2_api)
>>> destination = parse_sync_folder(destination, b2_api)

>>> policies_manager = ScanPoliciesManager(exclude_all_symlinks=True)

>>> synchronizer = Synchronizer(
        max_workers=10,
        policies_manager=policies_manager,
        dry_run=False,
        allow_empty_source=True,
        compare_version_mode=CompareVersionMode.SIZE,
        compare_threshold=10,
        newer_file_mode=NewerFileSyncMode.REPLACE,
        keep_days_or_delete=KeepOrDeleteMode.KEEP_BEFORE_DELETE,
        keep_days=10,
    )
```

We have a file (hello.txt) which is present in destination but not on source (my local), so it will be deleted and since our mode is to keep the delete file, it will be hidden for 10 days in bucket.

```
>>> no_progress = False
>>> with SyncReport(sys.stdout, no_progress) as reporter:
        synchronizer.sync_folders(
            source_folder=source,
            dest_folder=destination,
            now_millis=int(round(time.time() * 1000)),
            reporter=reporter,
        )
upload f1.txt
delete hello.txt (old version)
hide   hello.txt
```

We changed f1.txt and added 1 byte. Since our compare_threshold is 10, it will not do anything.

```
>>> with SyncReport(sys.stdout, no_progress) as reporter:
        synchronizer.sync_folders(
            source_folder=source,
            dest_folder=destination,
            now_millis=int(round(time.time() * 1000)),
            reporter=reporter,
        )
```

We changed f1.txt and added more than 10 bytes. Since our compare_threshold is 10, it will replace the file at destination folder.

```
>>> with SyncReport(sys.stdout, no_progress) as reporter:
        synchronizer.sync_folders(
            source_folder=source,
            dest_folder=destination,
            now_millis=int(round(time.time() * 1000)),
            reporter=reporter,
        )
upload f1.txt
```

Let's just delete the file and not keep - keep_days_or_delete = DELETE You can avoid passing keep_days argument in this case because it will be ignored anyways

```
>>> synchronizer = Synchronizer(
        max_workers=10,
        policies_manager=policies_manager,
        dry_run=False,
        allow_empty_source=True,
        compare_version_mode=CompareVersionMode.SIZE,
        compare_threshold=10,  # in bytes
        newer_file_mode=NewerFileSyncMode.REPLACE,
        keep_days_or_delete=KeepOrDeleteMode.DELETE,
    )

>>> with SyncReport(sys.stdout, no_progress) as reporter:
    synchronizer.sync_folders(
        source_folder=source,
        dest_folder=destination,
        now_millis=int(round(time.time() * 1000)),
        reporter=reporter,
    )
delete f1.txt
delete f1.txt (old version)
delete hello.txt (old version)
upload f2.txt
delete hello.txt (hide marker)
```

As you can see, it deleted f1.txt and it's older versions (no hide this time) and deleted hello.txt also because now we don't want the file anymore. also, we added another file f2.txt which gets uploaded.

Now we changed newer_file_mode to SKIP and compare_version_mode to MODTIME. also uploaded a new version of f2.txt to bucket using B2 web.

```
>>> synchronizer = Synchronizer(
        max_workers=10,
        policies_manager=policies_manager,
```

(continues on next page)

(continued from previous page)

```
        dry_run=False,
        allow_empty_source=True,
        compare_version_mode=CompareVersionMode.MODTIME,
        compare_threshold=10,  # in seconds
        newer_file_mode=NewerFileSyncMode.SKIP,
        keep_days_or_delete=KeepOrDeleteMode.DELETE,
    )
>>> with SyncReport(sys.stdout, no_progress) as reporter:
    synchronizer.sync_folders(
        source_folder=source,
        dest_folder=destination,
        now_millis=int(round(time.time() * 1000)),
        reporter=reporter,
    )
```

As expected, nothing happened, it found a file that was older at source but did not do anything because we skipped.

Now we changed newer_file_mode again to REPLACE and also uploaded a new version of f2.txt to bucket using B2 web.

```
>>> synchronizer = Synchronizer(
        max_workers=10,
        policies_manager=policies_manager,
        dry_run=False,
        allow_empty_source=True,
        compare_version_mode=CompareVersionMode.MODTIME,
        compare_threshold=10,
        newer_file_mode=NewerFileSyncMode.REPLACE,
        keep_days_or_delete=KeepOrDeleteMode.DELETE,
    )
>>> with SyncReport(sys.stdout, no_progress) as reporter:
    synchronizer.sync_folders(
        source_folder=source,
        dest_folder=destination,
        now_millis=int(round(time.time() * 1000)),
        reporter=reporter,
    )
delete f2.txt (old version)
upload f2.txt
```

class b2sdk.v1.**Synchronizer**

__init__(*max_workers*, *policies_manager=<b2sdk.sync.scan_policies.ScanPoliciesManager object>*, *dry_run=False*, *allow_empty_source=False*, *newer_file_mode=<NewerFileSyncMode.RAISE_ERROR: 103>*, *keep_days_or_delete=<KeepOrDeleteMode.NO_DELETE: 303>*, *compare_version_mode=<CompareVersionMode.MODTIME: 201>*, *compare_threshold=None*, *keep_days=None*)

Initialize synchronizer class and validate arguments

Parameters

- **max_workers** (*int*) – max number of workers
- **policies_manager** – policies manager object
- **dry_run** (*bool*) – test mode, does not actually transfer/delete when enabled
- **allow_empty_source** (*bool*) – if True, do not check whether source folder is empty

- **newer_file_mode** (*b2sdk.v1.NewerFileSyncMode*) – setting which determines handling for destination files newer than on the source
- **keep_days_or_delete** (*b2sdk.v1.KeepOrDeleteMode*) – setting which determines if we should delete or not delete or keep for *keep_days*
- **compare_version_mode** (*b2sdk.v1.CompareVersionMode*) – how to compare the source and destination files to find new ones
- **compare_threshold** (*int*) – should be greater than 0, default is 0
- **keep_days** (*int*) – if keep_days_or_delete is *b2sdk.v1.KeepOrDeleteMode.KEEP_BEFORE_DELETE*, then this should be greater than 0

sync_folders(*source_folder*, *dest_folder*, *now_millis*, *reporter*)

Syncs two folders. Always ensures that every file in the source is also in the destination. Deletes any file versions in the destination older than history_days.

Parameters

- **source_folder** (b2sdk.sync.folder.AbstractFolder) – source folder object
- **dest_folder** (b2sdk.sync.folder.AbstractFolder) – destination folder object
- **now_millis** (*int*) – current time in milliseconds
- **reporter** (*b2sdk.sync.report.SyncReport*, *None*) – progress reporter

make_folder_sync_actions(*source_folder*, *dest_folder*, *now_millis*, *reporter*, *policies_manager=<b2sdk.sync.scan_policies.ScanPoliciesManager object>*)

Yield a sequence of actions that will sync the destination folder to the source folder.

Parameters

- **source_folder** (*b2sdk.v1.AbstractFolder*) – source folder object
- **dest_folder** (*b2sdk.v1.AbstractFolder*) – destination folder object
- **now_millis** (*int*) – current time in milliseconds
- **reporter** (*b2sdk.v1.SyncReport*) – reporter object
- **policies_manager** – policies manager object

make_file_sync_actions(*sync_type*, *source_file*, *dest_file*, *source_folder*, *dest_folder*, *now_millis*)

Yields the sequence of actions needed to sync the two files

Parameters

- **sync_type** (*str*) – synchronization type
- **source_file** (*b2sdk.v1.File*) – source file object
- **dest_file** (*b2sdk.v1.File*) – destination file object
- **source_folder** (*b2sdk.v1.AbstractFolder*) – a source folder object
- **dest_folder** (*b2sdk.v1.AbstractFolder*) – a destination folder object
- **now_millis** (*int*) – current time in milliseconds

2.6.3 Internal API

Note: See *Internal interface* chapter to learn when and how to safely use the Internal API

b2sdk.session – B2 Session

class b2sdk.session.**B2Session**(*api*, *raw_api*)
Bases: object

A *magic* facade that supplies the correct api_url and account_auth_token to methods of underlying raw_api and reauthorizes if necessary.

__init__(*api*, *raw_api*)
Initialize self. See help(type(self)) for accurate signature.

b2sdk.raw_api – B2 raw api wrapper

class b2sdk.raw_api.**MetadataDirectiveMode**
Bases: enum.Enum

An enumeration.

COPY = 401

REPLACE = 402

class b2sdk.raw_api.**AbstractRawApi**
Bases: object

Direct access to the B2 web apis.

abstract cancel_large_file(*api_url*, *account_auth_token*, *file_id*)

abstract delete_bucket(*api_url*, *account_auth_token*, *account_id*, *bucket_id*)

abstract delete_file_version(*api_url*, *account_auth_token*, *file_id*, *file_name*)

abstract finish_large_file(*api_url*, *account_auth_token*, *file_id*, *part_sha1_array*)

abstract get_upload_part_url(*api_url*, *account_auth_token*, *file_id*)

abstract hide_file(*api_url*, *account_auth_token*, *bucket_id*, *file_name*)

abstract list_parts(*api_url*, *account_auth_token*, *file_id*, *start_part_number*, *max_part_count*)

abstract list_unfinished_large_files(*api_url*, *account_auth_token*, *bucket_id*, *start_file_id=None*, *max_file_count=None*)

abstract start_large_file(*api_url*, *account_auth_token*, *bucket_id*, *file_name*, *content_type*, *file_info*)

abstract update_bucket(*api_url*, *account_auth_token*, *account_id*, *bucket_id*, *bucket_type=None*, *bucket_info=None*, *cors_rules=None*, *lifecycle_rules=None*, *if_revision_is=None*)

abstract upload_part(*upload_url*, *upload_auth_token*, *part_number*, *content_length*, *sha1_sum*, *input_stream*)

get_download_url_by_id(*download_url*, *account_auth_token*, *file_id*)

get_download_url_by_name(*download_url*, *account_auth_token*, *bucket_name*, *file_name*)

abstract copy_file(*api_url*, *account_auth_token*, *source_file_id*, *new_file_name*, *bytes_range=None*, *metadata_directive=None*, *content_type=None*, *file_info=None*, *destination_bucket_id=None*)

class b2sdk.raw_api.**B2RawApi**(*b2_http*)

Bases: *b2sdk.raw_api.AbstractRawApi*

Provide access to the B2 web APIs, exactly as they are provided by b2.

Requires that you provide all necessary URLs and auth tokens for each call.

Each API call decodes the returned JSON and returns a dict.

For details on what each method does, see the B2 docs: https://www.backblaze.com/b2/docs/

This class is intended to be a super-simple, very thin layer on top of the HTTP calls. It can be mocked-out for testing higher layers. And this class can be tested by exercising each call just once, which is relatively quick.

All public methods of this class except authorize_account shall accept api_url and account_info as first two positional arguments. This is needed for B2Session magic.

__init__(*b2_http*)

Initialize self. See help(type(self)) for accurate signature.

authorize_account(*realm_url*, *application_key_id*, *application_key*)

cancel_large_file(*api_url*, *account_auth_token*, *file_id*)

create_bucket(*api_url*, *account_auth_token*, *account_id*, *bucket_name*, *bucket_type*, *bucket_info=None*, *cors_rules=None*, *lifecycle_rules=None*)

create_key(*api_url*, *account_auth_token*, *account_id*, *capabilities*, *key_name*, *valid_duration_seconds*, *bucket_id*, *name_prefix*)

delete_bucket(*api_url*, *account_auth_token*, *account_id*, *bucket_id*)

delete_file_version(*api_url*, *account_auth_token*, *file_id*, *file_name*)

delete_key(*api_url*, *account_auth_token*, *application_key_id*)

download_file_from_url(*_*, *account_auth_token_or_none*, *url*, *range_=None*)

Issue a streaming request for download of a file, potentially authorized.

Parameters

- *_* – unused (caused by B2Session magic)
- **account_auth_token_or_none** – an optional account auth token to pass in
- **url** – the full URL to download from
- **range** – two-element tuple for http Range header

Returns b2_http response

finish_large_file(*api_url*, *account_auth_token*, *file_id*, *part_sha1_array*)

get_download_authorization(*api_url*, *account_auth_token*, *bucket_id*, *file_name_prefix*, *valid_duration_in_seconds*)

get_file_info(*api_url*, *account_auth_token*, *file_id*)

get_upload_url(*api_url*, *account_auth_token*, *bucket_id*)

get_upload_part_url(*api_url*, *account_auth_token*, *file_id*)

hide_file(*api_url*, *account_auth_token*, *bucket_id*, *file_name*)

list_buckets(*api_url*, *account_auth_token*, *account_id*, *bucket_id=None*, *bucket_name=None*)

list_file_names(*api_url*, *account_auth_token*, *bucket_id*, *start_file_name=None*, *max_file_count=None*, *prefix=None*)

list_file_versions(*api_url*, *account_auth_token*, *bucket_id*, *start_file_name=None*, *start_file_id=None*, *max_file_count=None*, *prefix=None*)

list_keys(*api_url*, *account_auth_token*, *account_id*, *max_key_count=None*, *start_application_key_id=None*)

list_parts(*api_url*, *account_auth_token*, *file_id*, *start_part_number*, *max_part_count*)

list_unfinished_large_files(*api_url*, *account_auth_token*, *bucket_id*, *start_file_id=None*, *max_file_count=None*)

start_large_file(*api_url*, *account_auth_token*, *bucket_id*, *file_name*, *content_type*, *file_info*)

update_bucket(*api_url*, *account_auth_token*, *account_id*, *bucket_id*, *bucket_type=None*, *bucket_info=None*, *cors_rules=None*, *lifecycle_rules=None*, *if_revision_is=None*)

unprintable_to_hex(*string*)

Replace unprintable chars in string with a hex representation.

Parameters **string** – an arbitrary string, possibly with unprintable characters.

Returns the string, with unprintable characters changed to hex (e.g., “”)

check_b2_filename(*filename*)

Raise an appropriate exception with details if the filename is unusable.

See https://www.backblaze.com/b2/docs/files.html for the rules.

Parameters **filename** – a proposed filename in unicode

Returns None if the filename is usable

upload_file(*upload_url*, *upload_auth_token*, *file_name*, *content_length*, *content_type*, *content_sha1*, *file_infos*, *data_stream*)

Upload one, small file to b2.

Parameters

- **upload_url** – the upload_url from b2_authorize_account
- **upload_auth_token** – the auth token from b2_authorize_account
- **file_name** – the name of the B2 file
- **content_length** – number of bytes in the file
- **content_type** – MIME type
- **content_sha1** – hex SHA1 of the contents of the file
- **file_infos** – extra file info to upload
- **data_stream** – a file like object from which the contents of the file can be read

Returns

upload_part(*upload_url*, *upload_auth_token*, *part_number*, *content_length*, *content_sha1*, *data_stream*)

copy_file(*api_url*, *account_auth_token*, *source_file_id*, *new_file_name*, *bytes_range=None*, *metadata_directive=None*, *content_type=None*, *file_info=None*, *destination_bucket_id=None*)

b2sdk.raw_api.**test_raw_api**()

Exercise the code in B2RawApi by making each call once, just to make sure the parameters are passed in, and the result is passed back.

The goal is to be a complete test of B2RawApi, so the tests for the rest of the code can use the simulator.

Prints to stdout if things go wrong.

Returns 0 on success, non-zero on failure

b2sdk.raw_api.**test_raw_api_helper**(*raw_api*)
Try each of the calls to the raw api. Raise an exception if anything goes wrong.

This uses a Backblaze account that is just for this test. The account uses the free level of service, which should be enough to run this test a reasonable number of times each day. If somebody abuses the account for other things, this test will break and we'll have to do something about it.

b2sdk.b2http – thin http client wrapper

class b2sdk.b2http.**ResponseContextManager**(*response*)
A context manager that closes a requests.Response when done.

class b2sdk.b2http.**HttpCallback**
A callback object that does nothing. Overrides pre_request and/or post_request as desired.

pre_request(*method*, *url*, *headers*)
Called before processing an HTTP request.

Raises an exception if this request should not be processed. The exception raised must inherit from B2HttpCallbackPreRequestException.

Parameters

- **method** (*str*) – str, one of: 'POST', 'GET', etc.
- **url** (*str*) – the URL that will be used
- **headers** (*dict*) – the header sent with the request

post_request(*method*, *url*, *headers*, *response*)
Called after processing an HTTP request. Should not raise an exception.

Raises an exception if this request should be treated as failing. The exception raised must inherit from B2HttpCallbackPostRequestException.

Parameters

- **method** (*str*) – one of: 'POST', 'GET', etc.
- **url** (*str*) – the URL that will be used
- **headers** (*dict*) – the header sent with the request
- **response** – a response object from the requests library

class b2sdk.b2http.**ClockSkewHook**

post_request(*method*, *url*, *headers*, *response*)
Raise an exception if the clock in the server is too different from the clock on the local host.

The Date header contains a string that looks like: "Fri, 16 Dec 2016 20:52:30 GMT".

Parameters

- **method** (*str*) – one of: 'POST', 'GET', etc.
- **url** (*str*) – the URL that will be used
- **headers** (*dict*) – the header sent with the request

- **response** – a response object from the requests library

class b2sdk.b2http.**B2Http**(*requests_module=None*, *install_clock_skew_hook=True*)

A wrapper for the requests module. Provides the operations needed to access B2, and handles retrying when the returned status is 503 Service Unavailable, 429 Too Many Requests, etc.

The operations supported are:

- post_json_return_json
- post_content_return_json
- get_content

The methods that return JSON either return a Python dict or raise a subclass of B2Error. They can be used like this:

```
try:
    response_dict = b2_http.post_json_return_json(url, headers, params)
    ...
except B2Error as e:
    ...
```

TIMEOUT = 130

add_callback(*callback*)

Add a callback that inherits from HttpCallback.

Parameters **callback** (*callable*) – a callback to be added to a chain

post_content_return_json(*url*, *headers*, *data*, *try_count=5*, *post_params=None*)

Use like this:

```
try:
    response_dict = b2_http.post_content_return_json(url, headers, data)
    ...
except B2Error as e:
    ...
```

Parameters

- **url** (*str*) – a URL to call
- **headers** (*dict*) – headers to send.
- **data** – bytes (Python 3) or str (Python 2), or a file-like object, to send

Returns a dict that is the decoded JSON

Return type dict

post_json_return_json(*url*, *headers*, *params*, *try_count=5*)

Use like this:

```
try:
    response_dict = b2_http.post_json_return_json(url, headers, params)
    ...
except B2Error as e:
    ...
```

Parameters

- **url** (*str*) – a URL to call
- **headers** (*dict*) – headers to send.
- **params** (*dict*) – a dict that will be converted to JSON

Returns the decoded JSON document

Return type dict

get_content(*url*, *headers*, *try_count=5*)

Fetche content from a URL.

Use like this:

```
try:
    with b2_http.get_content(url, headers) as response:
        for byte_data in response.iter_content(chunk_size=1024):
            ...
except B2Error as e:
    ...
```

The response object is only guarantee to have:

- headers
- iter_content()

Parameters

- **url** (*str*) – a URL to call
- **headers** (*dict*) – headers to send
- **try_count** (*int*) – a number or retries

Returns Context manager that returns an object that supports iter_content()

b2sdk.b2http.**test_http**()

Run a few tests on error diagnosis.

This test takes a while to run and is not used in the automated tests during building. Run the test by hand to exercise the code. Be sure to run in both Python 2 and Python 3.

b2sdk.utils

b2sdk.utils.**interruptible_get_result**(*future*)

Wait for the result of a future in a way that can be interrupted by a KeyboardInterrupt.

This is not necessary in Python 3, but is needed for Python 2.

Parameters **future** (*Future*) – a future to get result of

b2sdk.utils.**b2_url_encode**(*s*)

URL-encode a unicode string to be sent to B2 in an HTTP header.

Parameters **s** (*str*) – a unicode string to encode

Returns URL-encoded string

Return type str

b2sdk.utils.**b2_url_decode**(*s*)
Decode a Unicode string returned from B2 in an HTTP header.

Parameters **s** (*str*) – a unicode string to decode

Returns a Python unicode string.

Return type str

b2sdk.utils.**choose_part_ranges**(*content_length*, *minimum_part_size*)
Return a list of (offset, length) for the parts of a large file.

Parameters

- **content_length** (*int*) – content length value
- **minimum_part_size** (*int*) – a minimum file part size

Return type list

b2sdk.utils.**hex_sha1_of_stream**(*input_stream*, *content_length*)
Return the 40-character hex SHA1 checksum of the first content_length bytes in the input stream.

Parameters

- **input_stream** – stream object, which exposes read() method
- **content_length** (*int*) – expected length of the stream

Return type str

b2sdk.utils.**hex_sha1_of_bytes**(*data*)
Return the 40-character hex SHA1 checksum of the data.

Parameters **data** (*bytes*) – an array of bytes

Return type str

b2sdk.utils.**validate_b2_file_name**(*name*)
Raise a ValueError if the name is not a valid B2 file name.

Parameters **name** (*str*) – a string to check

b2sdk.utils.**is_file_readable**(*local_path*, *reporter=None*)
Check if the local file has read permissions.

Parameters

- **local_path** (*str*) – a file path
- **reporter** – reporter object to put errors on

Return type bool

b2sdk.utils.**fix_windows_path_limit**(*path*)
Prefix paths when running on Windows to overcome 260 character path length limit. See https://msdn.microsoft.com/en-us/library/windows/desktop/aa365247(v=vs.85).aspx#maxpath

Parameters **path** (*str*) – a path to prefix

Returns a prefixed path

Return type str

class b2sdk.utils.**BytesIoContextManager**(*byte_data*)
Bases: object

A simple wrapper for a BytesIO that makes it look like a file-like object that can be a context manager.

__init__(*byte_data*)

Parameters **bytes_data** – a byte stream

class b2sdk.utils.**TempDir**
Bases: object

Context manager that creates and destroys a temporary directory.

b2sdk.utils.**format_and_scale_number**(*x*, *unit*)
Pick a good scale for representing a number and format it.

Parameters

- **x** (*int*) – a number
- **unit** (*str*) – an arbitrary unit name

Returns scaled and formatted number

Return type str

b2sdk.utils.**format_and_scale_fraction**(*numerator*, *denominator*, *unit*)
Pick a good scale for representing a fraction, and format it.

Parameters

- **numerator** (*int*) – a numerator of a fraction
- **denominator** (*int*) – a denominator of a fraction
- **unit** (*str*) – an arbitrary unit name

Returns scaled and formatted fraction

Return type str

b2sdk.utils.**camelcase_to_underscore**(*input_*)
Convert a camel-cased string to a string with underscores.

Parameters **input** (*str*) – an input string

Returns string with underscores

Return type str

class b2sdk.utils.**B2TraceMeta**
Bases: logfury.v0_1.meta.DefaultTraceMeta

Trace all public method calls, except for ones with names that begin with *get_*.

class b2sdk.utils.**B2TraceMetaAbstract**
Bases: logfury.v0_1.meta.DefaultTraceAbstractMeta

Default class for tracers, to be set as a metaclass for abstract base classes.

b2sdk.cache

class b2sdk.cache.**AbstractCache**
Bases: object

clear()

abstract get_bucket_id_or_none_from_bucket_name(*name*)

abstract get_bucket_name_or_none_from_allowed()

abstract save_bucket(*bucket*)

abstract set_bucket_name_cache(*buckets*)

class b2sdk.cache.**DummyCache**
Bases: *b2sdk.cache.AbstractCache*

A cache that does nothing.

get_bucket_id_or_none_from_bucket_name(*name*)

get_bucket_name_or_none_from_allowed()

save_bucket(*bucket*)

set_bucket_name_cache(*buckets*)

class b2sdk.cache.**InMemoryCache**
Bases: *b2sdk.cache.AbstractCache*

A cache that stores the information in memory.

__init__()
Initialize self. See help(type(self)) for accurate signature.

get_bucket_id_or_none_from_bucket_name(*name*)

get_bucket_name_or_none_from_allowed()

save_bucket(*bucket*)

set_bucket_name_cache(*buckets*)

class b2sdk.cache.**AuthInfoCache**(*info*)
Bases: *b2sdk.cache.AbstractCache*

A cache that stores data persistently in StoredAccountInfo.

__init__(*info*)
Initialize self. See help(type(self)) for accurate signature.

get_bucket_id_or_none_from_bucket_name(*name*)

get_bucket_name_or_none_from_allowed()

save_bucket(*bucket*)

set_bucket_name_cache(*buckets*)

b2sdk.download_dest – Download destination

class b2sdk.download_dest.**AbstractDownloadDestination**
Bases: object

Interface to a destination for a downloaded file.

abstract make_file_context(*file_id*, *file_name*, *content_length*, *content_type*, *content_sha1*, *file_info*, *mod_time_millis*, *range_=None*)
Return a context manager that yields a binary file-like object to use for writing the contents of the file.

Parameters

- **file_id** (*str*) – the B2 file ID from the headers
- **file_name** (*str*) – the B2 file name from the headers
- **content_type** (*str*) – the content type from the headers

- **content_sha1** (*str*) – the content sha1 from the headers (or "none" for large files)
- **file_info** (*dict*) – the user file info from the headers
- **mod_time_millis** (*int*) – the desired file modification date in ms since 1970-01-01
- **range** (*None*, *tuple[int, int]*) – starting and ending offsets of the received file contents. Usually None, which means that the whole file is downloaded.

Returns None

class b2sdk.download_dest.**DownloadDestLocalFile**(*local_file_path*)
Bases: *b2sdk.download_dest.AbstractDownloadDestination*

Store a downloaded file into a local file and sets its modification time.

MODE = 'wb+'

__init__(*local_file_path*)
Initialize self. See help(type(self)) for accurate signature.

make_file_context(*file_id*, *file_name*, *content_length*, *content_type*, *content_sha1*, *file_info*, *mod_time_millis*, *range_=None*)
Return a context manager that yields a binary file-like object to use for writing the contents of the file.

Parameters

- **file_id** (*str*) – the B2 file ID from the headers
- **file_name** (*str*) – the B2 file name from the headers
- **content_type** (*str*) – the content type from the headers
- **content_sha1** (*str*) – the content sha1 from the headers (or "none" for large files)
- **file_info** (*dict*) – the user file info from the headers
- **mod_time_millis** (*int*) – the desired file modification date in ms since 1970-01-01
- **range** (*None*, *tuple[int, int]*) – starting and ending offsets of the received file contents. Usually None, which means that the whole file is downloaded.

Returns None

write_to_local_file_context(*mod_time_millis*)

class b2sdk.download_dest.**PreSeekedDownloadDest**(*local_file_path*, *seek_target*)
Bases: *b2sdk.download_dest.DownloadDestLocalFile*

Store a downloaded file into a local file and sets its modification time. Does not truncate the target file, seeks to a given offset just after opening a descriptor.

MODE = 'rb+'

__init__(*local_file_path*, *seek_target*)
Initialize self. See help(type(self)) for accurate signature.

write_to_local_file_context(**args*, ***kwargs*)

class b2sdk.download_dest.**DownloadDestBytes**
Bases: *b2sdk.download_dest.AbstractDownloadDestination*

Store a downloaded file into bytes in memory.

__init__()
Initialize self. See help(type(self)) for accurate signature.

make_file_context(*file_id*, *file_name*, *content_length*, *content_type*, *content_sha1*, *file_info*, *mod_time_millis*, *range_=None*)
Return a context manager that yields a binary file-like object to use for writing the contents of the file.

Parameters

- **file_id** (*str*) – the B2 file ID from the headers
- **file_name** (*str*) – the B2 file name from the headers
- **content_type** (*str*) – the content type from the headers
- **content_sha1** (*str*) – the content sha1 from the headers (or "none" for large files)
- **file_info** (*dict*) – the user file info from the headers
- **mod_time_millis** (*int*) – the desired file modification date in ms since 1970-01-01
- **range** (*None*, *tuple[int, int]*) – starting and ending offsets of the received file contents. Usually None, which means that the whole file is downloaded.

Returns None

capture_bytes_context()
Remember the bytes written in self.bytes_written.

get_bytes_written()

class b2sdk.download_dest.**DownloadDestProgressWrapper**(*download_dest*, *progress_listener*)
Bases: *b2sdk.download_dest.AbstractDownloadDestination*

Wrap a DownloadDestination and report progress to a ProgressListener.

__init__(*download_dest*, *progress_listener*)
Initialize self. See help(type(self)) for accurate signature.

make_file_context(*file_id*, *file_name*, *content_length*, *content_type*, *content_sha1*, *file_info*, *mod_time_millis*, *range_=None*)
Return a context manager that yields a binary file-like object to use for writing the contents of the file.

Parameters

- **file_id** (*str*) – the B2 file ID from the headers
- **file_name** (*str*) – the B2 file name from the headers
- **content_type** (*str*) – the content type from the headers
- **content_sha1** (*str*) – the content sha1 from the headers (or "none" for large files)
- **file_info** (*dict*) – the user file info from the headers
- **mod_time_millis** (*int*) – the desired file modification date in ms since 1970-01-01
- **range** (*None*, *tuple[int, int]*) – starting and ending offsets of the received file contents. Usually None, which means that the whole file is downloaded.

Returns None

write_file_and_report_progress_context(*file_id*, *file_name*, *content_length*, *content_type*, *content_sha1*, *file_info*, *mod_time_millis*, *range_*)

b2sdk.sync.action

class b2sdk.sync.action.**AbstractAction**
Bases: object

An action to take, such as uploading, downloading, or deleting a file. Multi-threaded tasks create a sequence of Actions which are then run by a pool of threads.

An action can depend on other actions completing. An example of this is making sure a CreateBucketAction happens before an UploadFileAction.

run(*bucket*, *reporter*, *dry_run=False*)
Main action routine.

Parameters

- **bucket** (*b2sdk.bucket.Bucket*) – a Bucket object
- **reporter** – a place to report errors
- **dry_run** (*bool*) – if True, perform a dry run

abstract get_bytes()
Return the number of bytes to transfer for this action.

Return type int

abstract do_action(*bucket*, *reporter*)
Perform the action, returning only after the action is completed.

Parameters

- **bucket** (*b2sdk.bucket.Bucket*) – a Bucket object
- **reporter** – a place to report errors

abstract do_report(*bucket*, *reporter*)
Report the action performed.

Parameters

- **bucket** (*b2sdk.bucket.Bucket*) – a Bucket object
- **reporter** – a place to report errors

class b2sdk.sync.action.**B2UploadAction**(*local_full_path*, *relative_name*, *b2_file_name*, *mod_time_millis*, *size*)
Bases: *b2sdk.sync.action.AbstractAction*

File uploading action.

__init__(*local_full_path*, *relative_name*, *b2_file_name*, *mod_time_millis*, *size*)

Parameters

- **local_full_path** (*str*) – a local file path
- **relative_name** (*str*) – a relative file name
- **b2_file_name** (*str*) – a name of a new remote file
- **mod_time_millis** (*int*) – file modification time in milliseconds
- **size** (*int*) – a file size

get_bytes()
Return file size.

Return type int

do_action(*bucket*, *reporter*)
Perform the uploading action, returning only after the action is completed.

Parameters

- **bucket** (*b2sdk.bucket.Bucket*) – a Bucket object
- **reporter** – a place to report errors

do_report(*bucket*, *reporter*)
Report the uploading action performed.

Parameters

- **bucket** (*b2sdk.bucket.Bucket*) – a Bucket object
- **reporter** – a place to report errors

class b2sdk.sync.action.**B2HideAction**(*relative_name*, *b2_file_name*)
Bases: *b2sdk.sync.action.AbstractAction*

__init__(*relative_name*, *b2_file_name*)

Parameters

- **relative_name** (*str*) – a relative file name
- **b2_file_name** (*str*) – a name of a remote file

get_bytes()
Return file size.

Returns always zero

Return type int

do_action(*bucket*, *reporter*)
Perform the hiding action, returning only after the action is completed.

Parameters

- **bucket** (*b2sdk.bucket.Bucket*) – a Bucket object
- **reporter** – a place to report errors

do_report(*bucket*, *reporter*)
Report the hiding action performed.

Parameters

- **bucket** (*b2sdk.bucket.Bucket*) – a Bucket object
- **reporter** – a place to report errors

class b2sdk.sync.action.**B2DownloadAction**(*relative_name*, *b2_file_name*, *file_id*, *local_full_path*, *mod_time_millis*, *file_size*)
Bases: *b2sdk.sync.action.AbstractAction*

__init__(*relative_name*, *b2_file_name*, *file_id*, *local_full_path*, *mod_time_millis*, *file_size*)

Parameters

- **relative_name** (*str*) – a relative file name
- **b2_file_name** (*str*) – a name of a remote file
- **file_id** (*str*) – a file ID

- **local_full_path** (*str*) – a local file path
- **mod_time_millis** (*int*) – file modification time in milliseconds
- **file_size** (*int*) – a file size

get_bytes()
Return file size.

Return type int

do_action(*bucket*, *reporter*)
Perform the downloading action, returning only after the action is completed.

Parameters

- **bucket** (*b2sdk.bucket.Bucket*) – a Bucket object
- **reporter** – a place to report errors

do_report(*bucket*, *reporter*)
Report the downloading action performed.

Parameters

- **bucket** (*b2sdk.bucket.Bucket*) – a Bucket object
- **reporter** – a place to report errors

class b2sdk.sync.action.**B2DeleteAction**(*relative_name*, *b2_file_name*, *file_id*, *note*)
Bases: *b2sdk.sync.action.AbstractAction*

__init__(*relative_name*, *b2_file_name*, *file_id*, *note*)

Parameters

- **relative_name** (*str*) – a relative file name
- **b2_file_name** (*str*) – a name of a remote file
- **file_id** (*str*) – a file ID
- **note** (*str*) – a deletion note

get_bytes()
Return file size.

Returns always zero

Return type int

do_action(*bucket*, *reporter*)
Perform the deleting action, returning only after the action is completed.

Parameters

- **bucket** (*b2sdk.bucket.Bucket*) – a Bucket object
- **reporter** – a place to report errors

do_report(*bucket*, *reporter*)
Report the deleting action performed.

Parameters

- **bucket** (*b2sdk.bucket.Bucket*) – a Bucket object
- **reporter** – a place to report errors

class b2sdk.sync.action.**LocalDeleteAction**(*relative_name*, *full_path*)
Bases: *b2sdk.sync.action.AbstractAction*

__init__(*relative_name*, *full_path*)

Parameters

- **relative_name** (*str*) – a relative file name
- **full_path** – a full local path

Type str

get_bytes()
Return file size.

Returns always zero

Return type int

do_action(*bucket*, *reporter*)
Perform the deleting of a local file action, returning only after the action is completed.

Parameters

- **bucket** (*b2sdk.bucket.Bucket*) – a Bucket object
- **reporter** – a place to report errors

do_report(*bucket*, *reporter*)
Report the deleting of a local file action performed.

Parameters

- **bucket** (*b2sdk.bucket.Bucket*) – a Bucket object
- **reporter** – a place to report errors

b2sdk.sync.exception

exception b2sdk.sync.exception.**EnvironmentEncodingError**(*filename*, *encoding*)
Bases: b2sdk.exception.B2Error

Raised when a file name can not be decoded with system encoding.

__init__(*filename*, *encoding*)

Parameters

- **filename** (*str, bytes*) – an encoded file name
- **encoding** (*str*) – file name encoding

exception b2sdk.sync.exception.**InvalidArgument**(*parameter_name*, *message*)
Bases: b2sdk.exception.B2Error

Raised when one or more arguments are invalid

__init__(*parameter_name*, *message*)

Parameters

- **parameter_name** – name of the function argument
- **message** – brief explanation of misconfiguration

exception b2sdk.sync.exception.**IncompleteSync**(**args*, ***kwargs*)
Bases: b2sdk.exception.B2SimpleError

b2sdk.sync.file

class b2sdk.sync.file.**File**(*name*, *versions*)
Bases: object

Hold information about one file in a folder.

The name is relative to the folder in all cases.

Files that have multiple versions (which only happens in B2, not in local folders) include information about all of the versions, most recent first.

__init__(*name*, *versions*)

Parameters

- **name** (*str*) – a relative file name
- **versions** (*list*) – a list of file versions

latest_version()
Return the latest file version.

class b2sdk.sync.file.**FileVersion**(*id_*, *file_name*, *mod_time*, *action*, *size*)
Bases: object

Hold information about one version of a file.

__init__(*id_*, *file_name*, *mod_time*, *action*, *size*)

Parameters

- **id** (*str*) – the B2 file id, or the local full path name
- **file_name** (*str*) – a relative file name
- **mod_time** (*int*) – modification time, in milliseconds, to avoid rounding issues with millisecond times from B2
- **action** (*str*) – "hide" or "upload" (never "start")
- **size** (*int*) – a file size

b2sdk.sync.folder

class b2sdk.sync.folder.**AbstractFolder**
Bases: object

Interface to a folder full of files, which might be a B2 bucket, a virtual folder in a B2 bucket, or a directory on a local file system.

Files in B2 may have multiple versions, while files in local folders have just one.

abstract all_files(*reporter*, *policies_manager*)
Return an iterator over all of the files in the folder, in the order that B2 uses.

It also performs filtering using policies manager.

No matter what the folder separator on the local file system is, "/" is used in the returned file names.

If a file is found, but does not exist (for example due to a broken symlink or a race), reporter will be informed about each such problem.

Parameters

- **reporter** – a place to report errors
- **policies_manager** – a policies manager object

abstract folder_type()
Return one of: 'b2', 'local'.

Return type str

abstract make_full_path(*file_name*)
Return the full path to the file; only for local folders.

Parameters **file_name** (*str*) – a file name

b2sdk.sync.folder.**join_b2_path**(*b2_dir*, *b2_name*)
Like os.path.join, but for B2 file names where the root directory is called ''.

Parameters

- **b2_dir** (*str*) – a directory path
- **b2_name** (*str*) – a file name

class b2sdk.sync.folder.**LocalFolder**(*root*)
Bases: *b2sdk.sync.folder.AbstractFolder*

Folder interface to a directory on the local machine.

__init__(*root*)
Initialize a new folder.

Parameters **root** (*str*) – path to the root of the local folder. Must be unicode.

folder_type()
Return folder type.

Return type str

all_files(*reporter*, *policies_manager=<b2sdk.sync.scan_policies.ScanPoliciesManager object>*)
Yield all files.

Parameters

- **reporter** – a place to report errors
- **policies_manager** – a policy manager object, default is DEFAULT_SCAN_MANAGER

make_full_path(*file_name*)
Convert a file name into an absolute path.

Parameters **file_name** (*str*) – a file name

ensure_present()
Make sure that the directory exists.

ensure_non_empty()
Make sure that the directory exists and is non-empty.

class b2sdk.sync.folder.**B2Folder**(*bucket_name*, *folder_name*, *api*)
Bases: *b2sdk.sync.folder.AbstractFolder*

Folder interface to b2.

__init__(*bucket_name*, *folder_name*, *api*)

Parameters

- **bucket_name** (*str*) – a name of the bucket
- **folder_name** (*str*) – a folder name
- **api** (*b2sdk.api.B2Api*) – an API object

all_files(*reporter*, *policies_manager=<b2sdk.sync.scan_policies.ScanPoliciesManager object>*)
Yield all files.

Parameters

- **reporter** – a place to report errors
- **policies_manager** – a policies manager object, default is DEFAULT_SCAN_MANAGER

folder_type()
Return folder type.

Return type str

make_full_path(*file_name*)
Make an absolute path from a file name.

Parameters **file_name** (*str*) – a file name

b2sdk.sync.folder_parser

b2sdk.sync.folder_parser.**parse_sync_folder**(*folder_name*, *api*)
Take either a local path, or a B2 path, and returns a Folder object for it.

B2 paths look like: b2://bucketName/path/name. The '//' is optional, because the previous sync command didn't use it.

Anything else is treated like a local folder.

Parameters

- **folder_name** (*str*) – a name of the folder, either local or remote
- **api** (*b2sdk.api.B2Api*) – an API object

b2sdk.sync.policy

class b2sdk.sync.policy.**NewerFileSyncMode**
Bases: enum.Enum

An enumeration.

SKIP = 101

REPLACE = 102

RAISE_ERROR = 103

class b2sdk.sync.policy.**CompareVersionMode**
Bases: enum.Enum

An enumeration.

MODTIME = 201

SIZE = 202

NONE = 203

class b2sdk.sync.policy.**AbstractFileSyncPolicy**(*source_file*, *source_folder*, *dest_file*, *dest_folder*, *now_millis*, *keep_days*, *newer_file_mode*, *compare_threshold*, *compare_version_mode=<CompareVersionMode.MODTIME: 201>*)

Bases: object

Abstract policy class.

DESTINATION_PREFIX = NotImplemented

SOURCE_PREFIX = NotImplemented

__init__(*source_file*, *source_folder*, *dest_file*, *dest_folder*, *now_millis*, *keep_days*, *newer_file_mode*, *compare_threshold*, *compare_version_mode=<CompareVersionMode.MODTIME: 201>*)

Parameters

- **source_file** (*b2sdk.v1.File*) – source file object
- **source_folder** (*b2sdk.v1.AbstractFolder*) – source folder object
- **dest_file** (*b2sdk.v1.File*) – destination file object
- **dest_folder** (*b2sdk.v1.AbstractFolder*) – destination folder object
- **now_millis** (*int*) – current time in milliseconds
- **keep_days** (*int*) – days to keep before delete
- **newer_file_mode** (*b2sdk.v1.NEWER_FILE_MODES*) – setting which determines handling for destination files newer than on the source
- **compare_threshold** (*int*) – when comparing with size or time for sync
- **compare_version_mode** (*b2sdk.v1.COMPARE_VERSION_MODES*) – how to compare source and destination files

classmethod files_are_different(*source_file*, *dest_file*, *compare_threshold=None*, *compare_version_mode=<CompareVersionMode.MODTIME: 201>*, *newer_file_mode=<NewerFileSyncMode.RAISE_ERROR: 103>*)

Compare two files and determine if the the destination file should be replaced by the source file.

Parameters

- **source_file** (*b2sdk.v1.File*) – source file object
- **dest_file** (*b2sdk.v1.File*) – destination file object
- **compare_threshold** (*int*) – compare threshold when comparing by time or size
- **compare_version_mode** (*b2sdk.v1.CompareVersionMode*) – source file version comparator method
- **newer_file_mode** (*b2sdk.v1.NewerFileSyncMode*) – newer destination handling method

get_all_actions()

Yield file actions.

class b2sdk.sync.policy.**DownPolicy**(*source_file*, *source_folder*, *dest_file*, *dest_folder*, *now_millis*, *keep_days*, *newer_file_mode*, *compare_threshold*, *compare_version_mode=<CompareVersionMode.MODTIME: 201>*)

Bases: *b2sdk.sync.policy.AbstractFileSyncPolicy*

File is synced down (from the cloud to disk).

DESTINATION_PREFIX = 'local://'

SOURCE_PREFIX = 'b2://'

class b2sdk.sync.policy.**UpPolicy**(*source_file*, *source_folder*, *dest_file*, *dest_folder*, *now_millis*, *keep_days*, *newer_file_mode*, *compare_threshold*, *compare_version_mode=<CompareVersionMode.MODTIME: 201>*)

Bases: *b2sdk.sync.policy.AbstractFileSyncPolicy*

File is synced up (from disk the cloud).

DESTINATION_PREFIX = 'b2://'

SOURCE_PREFIX = 'local://'

class b2sdk.sync.policy.**UpAndDeletePolicy**(*source_file*, *source_folder*, *dest_file*, *dest_folder*, *now_millis*, *keep_days*, *newer_file_mode*, *compare_threshold*, *compare_version_mode=<CompareVersionMode.MODTIME: 201>*)

Bases: *b2sdk.sync.policy.UpPolicy*

File is synced up (from disk to the cloud) and the delete flag is SET.

class b2sdk.sync.policy.**UpAndKeepDaysPolicy**(*source_file*, *source_folder*, *dest_file*, *dest_folder*, *now_millis*, *keep_days*, *newer_file_mode*, *compare_threshold*, *compare_version_mode=<CompareVersionMode.MODTIME: 201>*)

Bases: *b2sdk.sync.policy.UpPolicy*

File is synced up (from disk to the cloud) and the keepDays flag is SET.

class b2sdk.sync.policy.**DownAndDeletePolicy**(*source_file*, *source_folder*, *dest_file*, *dest_folder*, *now_millis*, *keep_days*, *newer_file_mode*, *compare_threshold*, *compare_version_mode=<CompareVersionMode.MODTIME: 201>*)

Bases: *b2sdk.sync.policy.DownPolicy*

File is synced down (from the cloud to disk) and the delete flag is SET.

class b2sdk.sync.policy.**DownAndKeepDaysPolicy**(*source_file*, *source_folder*, *dest_file*, *dest_folder*, *now_millis*, *keep_days*, *newer_file_mode*, *compare_threshold*, *compare_version_mode=<CompareVersionMode.MODTIME: 201>*)

Bases: *b2sdk.sync.policy.DownPolicy*

File is synced down (from the cloud to disk) and the keepDays flag is SET.

b2sdk.sync.policy.**make_b2_delete_note**(*version*, *index*, *transferred*)

Create a note message for delete action.

Parameters

- **version** (b2sdk.v1.FileVersionInfo) – an object which contains file version info
- **index** (*int*) – file version index
- **transferred** (*bool*) – if True, file has been transferred, False otherwise

b2sdk.sync.policy.**make_b2_delete_actions**(*source_file*, *dest_file*, *dest_folder*, *transferred*)

Create the actions to delete files stored on B2, which are not present locally.

Parameters

- **source_file** (*b2sdk.v1.File*) – source file object
- **dest_file** (*b2sdk.v1.File*) – destination file object
- **dest_folder** (*b2sdk.v1.AbstractFolder*) – destination folder
- **transferred** (*bool*) – if True, file has been transferred, False otherwise

b2sdk.sync.policy.**make_b2_keep_days_actions**(*source_file*, *dest_file*, *dest_folder*, *transferred*, *keep_days*, *now_millis*)

Create the actions to hide or delete existing versions of a file stored in b2.

When keepDays is set, all files that were visible any time from keepDays ago until now must be kept. If versions were uploaded 5 days ago, 15 days ago, and 25 days ago, and the keepDays is 10, only the 25-day old version can be deleted. The 15 day-old version was visible 10 days ago.

Parameters

- **source_file** (*b2sdk.v1.File*) – source file object
- **dest_file** (*b2sdk.v1.File*) – destination file object
- **dest_folder** (*b2sdk.v1.AbstractFolder*) – destination folder object
- **transferred** (*bool*) – if True, file has been transferred, False otherwise
- **keep_days** (*int*) – how many days to keep a file
- **now_millis** (*int*) – current time in milliseconds

b2sdk.sync.policy_manager

class b2sdk.sync.policy_manager.**SyncPolicyManager**

Bases: object

Policy manager; implement a logic to get a correct policy class and create a policy object based on various parameters.

__init__()

Initialize self. See help(type(self)) for accurate signature.

get_policy(*sync_type*, *source_file*, *source_folder*, *dest_file*, *dest_folder*, *now_millis*, *delete*, *keep_days*, *newer_file_mode*, *compare_threshold*, *compare_version_mode*)

Return a policy object.

Parameters

- **sync_type** (*str*) – synchronization type
- **source_file** (*str*) – source file name
- **source_folder** (*str*) – a source folder path
- **dest_file** (*str*) – destination file name

- **dest_folder** (*str*) – a destination folder path
- **now_millis** (*int*) – current time in milliseconds
- **delete** (*bool*) – delete policy
- **keep_days** (*int*) – keep for days policy
- **newer_file_mode** (*b2sdk.v1.NewerFileSyncMode*) – setting which determines handling for destination files newer than on the source
- **compare_threshold** (*int*) – difference between file modification time or file size
- **compare_version_mode** (*b2sdk.v1.CompareVersionMode*) – setting which determines how to compare source and destination files

Returns a policy object

get_policy_class(*sync_type*, *delete*, *keep_days*)
Get policy class by a given sync type.

Parameters

- **sync_type** (*str*) – synchronization type
- **delete** (*bool*) – if True, delete files and update from source
- **keep_days** (*int*) – keep for *keep_days* before delete

Returns a policy class

b2sdk.sync.scan_policies

class b2sdk.sync.scan_policies.**RegexSet**(*regex_iterable*)
Bases: object

Hold a (possibly empty) set of regular expressions and know how to check whether a string matches any of them.

__init__(*regex_iterable*)

Parameters **regex_iterable** – an interable which yields regexes

matches(*s*)
Check whether a string matches any of regular expressions.

Parameters **s** (*str*) – a string to check

Return type bool

b2sdk.sync.scan_policies.**convert_dir_regex_to_dir_prefix_regex**(*dir_regex*)
The patterns used to match directory names (and file names) are allowed to match a prefix of the name. This 'feature' was unintentional, but is being retained for compatibility.

This means that a regex that matches a directory name can't be used directly to match against a file name and test whether the file should be excluded because it matches the directory.

The pattern 'photos' will match directory names 'photos' and 'photos2', and should exclude files 'photos/kitten.jpg', and 'photos2/puppy.jpg'. It should not exclude 'photos.txt', because there is no directory name that matches.

On the other hand, the pattern 'photos$' should match 'photos/kitten.jpg', but not 'photos2/puppy.jpg', nor 'photos.txt'

If the original regex is valid, there are only two cases to consider: either the regex ends in '$' or does not.

Parameters **dir_regex** (*str*) – a regular expression string or literal

class b2sdk.sync.scan_policies.**ScanPoliciesManager**(*exclude_dir_regexes=()*, *exclude_file_regexes=()*, *include_file_regexes=()*, *exclude_all_symlinks=False*)

Bases: object

Policy object used when scanning folders for syncing, used to decide which files to include in the list of files to be synced.

Code that scans through files should at least use should_exclude_file() to decide whether each file should be included; it will check include/exclude patterns for file names, as well as patterns for excluding directeries.

Code that scans may optionally use should_exclude_directory() to test whether it can skip a directory completely and not bother listing the files and sub-directories in it.

__init__(*exclude_dir_regexes=()*, *exclude_file_regexes=()*, *include_file_regexes=()*, *exclude_all_symlinks=False*)

Parameters

- **exclude_dir_regexes** (*tuple*) – a tuple of regexes to exclude directories
- **exclude_file_regexes** (*tuple*) – a tuple of regexes to exclude files
- **include_file_regexes** (*tuple*) – a tuple of regexes to include files
- **exclude_all_symlinks** (*bool*) – if True, exclude all symlinks

should_exclude_file(*file_path*)

Given the full path of a file, decide if it should be excluded from the scan.

Parameters **file_path** – the path of the file, relative to the root directory being scanned.

Type str

Returns True if excluded.

Return type bool

should_exclude_directory(*dir_path*)

Given the full path of a directory, decide if all of the files in it should be excluded from the scan.

Parameters **dir_path** (*str*) – the path of the directory, relative to the root directory being scanned. The path will never end in '/'.

Returns True if excluded.

b2sdk.sync.sync

b2sdk.sync.sync.**next_or_none**(*iterator*)

Return the next item from the iterator, or None if there are no more.

b2sdk.sync.sync.**zip_folders**(*folder_a*, *folder_b*, *reporter*, *policies_manager=<b2sdk.sync.scan_policies.ScanPoliciesManager object>*)

Iterate over all of the files in the union of two folders, matching file names.

Each item is a pair (file_a, file_b) with the corresponding file in both folders. Either file (but not both) will be None if the file is in only one folder.

Parameters

- **folder_a** (b2sdk.sync.folder.AbstractFolder) – first folder object.

- **folder_b** (b2sdk.sync.folder.AbstractFolder) – second folder object.
- **reporter** – reporter object
- **policies_manager** – policies manager object

Returns yields two element tuples

b2sdk.sync.sync.**count_files**(*local_folder*, *reporter*)
Count all of the files in a local folder.

Parameters

- **local_folder** (b2sdk.sync.folder.AbstractFolder) – a folder object.
- **reporter** – reporter object

class b2sdk.sync.sync.**KeepOrDeleteMode**
Bases: enum.Enum

An enumeration.

DELETE = 301

KEEP_BEFORE_DELETE = 302

NO_DELETE = 303

class b2sdk.sync.sync.**Synchronizer**(*max_workers*, *policies_manager=<b2sdk.sync.scan_policies.ScanPoliciesManager object>*, *dry_run=False*, *allow_empty_source=False*, *newer_file_mode=<NewerFileSyncMode.RAISE_ERROR: 103>*, *keep_days_or_delete=<KeepOrDeleteMode.NO_DELETE: 303>*, *compare_version_mode=<CompareVersionMode.MODTIME: 201>*, *compare_threshold=None*, *keep_days=None*)
Bases: object

__init__(*max_workers*, *policies_manager=<b2sdk.sync.scan_policies.ScanPoliciesManager object>*, *dry_run=False*, *allow_empty_source=False*, *newer_file_mode=<NewerFileSyncMode.RAISE_ERROR: 103>*, *keep_days_or_delete=<KeepOrDeleteMode.NO_DELETE: 303>*, *compare_version_mode=<CompareVersionMode.MODTIME: 201>*, *compare_threshold=None*, *keep_days=None*)
Initialize synchronizer class and validate arguments

Parameters

- **max_workers** (*int*) – max number of workers
- **policies_manager** – policies manager object
- **dry_run** (*bool*) – test mode, does not actually transfer/delete when enabled
- **allow_empty_source** (*bool*) – if True, do not check whether source folder is empty
- **newer_file_mode** (*b2sdk.v1.NewerFileSyncMode*) – setting which determines handling for destination files newer than on the source
- **keep_days_or_delete** (*b2sdk.v1.KeepOrDeleteMode*) – setting which determines if we should delete or not delete or keep for *keep_days*
- **compare_version_mode** (*b2sdk.v1.CompareVersionMode*) – how to compare the source and destination files to find new ones
- **compare_threshold** (*int*) – should be greater than 0, default is 0
- **keep_days** (*int*) – if keep_days_or_delete is *b2sdk.v1.KeepOrDeleteMode.KEEP_BEFORE_DELETE*, then this should be greater than 0

sync_folders(*source_folder*, *dest_folder*, *now_millis*, *reporter*)
Syncs two folders. Always ensures that every file in the source is also in the destination. Deletes any file versions in the destination older than history_days.

Parameters

- **source_folder** (b2sdk.sync.folder.AbstractFolder) – source folder object
- **dest_folder** (b2sdk.sync.folder.AbstractFolder) – destination folder object
- **now_millis** (*int*) – current time in milliseconds
- **reporter** (*b2sdk.sync.report.SyncReport*, *None*) – progress reporter

make_folder_sync_actions(*source_folder*, *dest_folder*, *now_millis*, *reporter*, *policies_manager=<b2sdk.sync.scan_policies.ScanPoliciesManager object>*)
Yield a sequence of actions that will sync the destination folder to the source folder.

Parameters

- **source_folder** (*b2sdk.v1.AbstractFolder*) – source folder object
- **dest_folder** (*b2sdk.v1.AbstractFolder*) – destination folder object
- **now_millis** (*int*) – current time in milliseconds
- **reporter** (*b2sdk.v1.SyncReport*) – reporter object
- **policies_manager** – policies manager object

make_file_sync_actions(*sync_type*, *source_file*, *dest_file*, *source_folder*, *dest_folder*, *now_millis*)
Yields the sequence of actions needed to sync the two files

Parameters

- **sync_type** (*str*) – synchronization type
- **source_file** (*b2sdk.v1.File*) – source file object
- **dest_file** (*b2sdk.v1.File*) – destination file object
- **source_folder** (*b2sdk.v1.AbstractFolder*) – a source folder object
- **dest_folder** (*b2sdk.v1.AbstractFolder*) – a destination folder object
- **now_millis** (*int*) – current time in milliseconds

b2sdk.transferer.abstract – Downloader base class

class b2sdk.transferer.abstract.**AbstractDownloader**(*force_chunk_size=None*, *min_chunk_size=None*, *max_chunk_size=None*)
Bases: object

__init__(*force_chunk_size=None*, *min_chunk_size=None*, *max_chunk_size=None*)
Initialize self. See help(type(self)) for accurate signature.

abstract is_suitable(*metadata*, *progress_listener*)
Analyze metadata (possibly against options passed earlier to constructor to find out whether the given download request should be handled by this downloader).

abstract download(*file*, *response*, *metadata*, *session*)
@returns (bytes_read, actual_sha1)

b2sdk.transferer.file_metadata

class b2sdk.transferer.file_metadata.**FileMetadata**(*file_id*, *file_name*, *content_type*, *content_length*, *content_sha1*, *file_info*)

Bases: object

Hold information about a file which is being downloaded.

__init__(*file_id*, *file_name*, *content_type*, *content_length*, *content_sha1*, *file_info*)
Initialize self. See help(type(self)) for accurate signature.

file_id

file_name

content_type

content_length

content_sha1

file_info

classmethod from_response(*response*)

as_info_dict()

b2sdk.transferer.parallel – ParallelTransferer

class b2sdk.transferer.parallel.**ParallelDownloader**(*max_streams*, *min_part_size*, **args*, ***kwargs*)

Bases: *b2sdk.transferer.abstract.AbstractDownloader*

FINISH_HASHING_BUFFER_SIZE = 1048576

__init__(*max_streams*, *min_part_size*, **args*, ***kwargs*)

Parameters

- **max_streams** – maximum number of simultaneous streams
- **min_part_size** – minimum amount of data a single stream will retrieve, in bytes

is_suitable(*metadata*, *progress_listener*)
Analyze metadata (possibly against options passed earlier to constructor to find out whether the given download request should be handled by this downloader).

download(*file*, *response*, *metadata*, *session*)
Download a file from given url using parallel download sessions and stores it in the given download_destination.

Parameters

- **file** – an opened file-like object to write to
- **response** – the response of the first request made to the cloud service with download intent

Returns

class b2sdk.transferer.parallel.**WriterThread**(*file*)

Bases: threading.Thread

__init__(*file*)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is the argument tuple for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.__init__()) before doing anything else to the thread.

run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

class b2sdk.transferer.parallel.**AbstractDownloaderThread**(*session*, *writer*, *part_to_download*, *chunk_size*)

Bases: threading.Thread

__init__(*session*, *writer*, *part_to_download*, *chunk_size*)

Parameters

- **session** – raw_api wrapper
- **writer** – where to write data
- **part_to_download** – PartToDownload object
- **chunk_size** – internal buffer size to use for writing and hashing

abstract run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

class b2sdk.transferer.parallel.**FirstPartDownloaderThread**(*response*, *hasher*, **args*, ***kwargs*)

Bases: *b2sdk.transferer.parallel.AbstractDownloaderThread*

__init__(*response*, *hasher*, **args*, ***kwargs*)

Parameters

- **response** – response of the original GET call
- **hasher** – hasher object to feed to as the stream is written

run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

class b2sdk.transferer.parallel.**NonHashingDownloaderThread**(*url*, **args*, ***kwargs*)
Bases: *b2sdk.transferer.parallel.AbstractDownloaderThread*

__init__(*url*, **args*, ***kwargs*)

Parameters **url** – url of the target file

run()
Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

class b2sdk.transferer.parallel.**PartToDownload**(*cloud_range*, *local_range*)
Bases: object

Hold the range of a file to download, and the range of the local file where it should be stored.

__init__(*cloud_range*, *local_range*)
Initialize self. See help(type(self)) for accurate signature.

b2sdk.transferer.parallel.**gen_parts**(*cloud_range*, *local_range*, *part_count*)
Generate a sequence of PartToDownload to download a large file as a collection of parts.

b2sdk.transferer.range – transfer range toolkit

class b2sdk.transferer.range.**Range**(*start*, *end*)
Bases: object

HTTP ranges use an *inclusive* index at the end.

__init__(*start*, *end*)
Initialize self. See help(type(self)) for accurate signature.

classmethod **from_header**(*raw_range_header*)
Factory method which returns an object constructed from Range http header.

raw_range_header example: 'bytes 0-11'

size()

subrange(*sub_start*, *sub_end*)
Return a range that is part of this range.

Parameters

- **sub_start** – index relative to the start of this range.
- **sub_end** – (Inclusive!) index relative to the start of this range.

Returns a new Range

as_tuple()

b2sdk.transferer.simple – SimpleDownloader

class b2sdk.transferer.simple.**SimpleDownloader**(**args*, ***kwargs*)
Bases: *b2sdk.transferer.abstract.AbstractDownloader*

__init__(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

is_suitable(*metadata*, *progress_listener*)
Analyze metadata (possibly against options passed earlier to constructor to find out whether the given download request should be handled by this downloader).

download(*file*, *response*, *metadata*, *session*)
@returns (bytes_read, actual_sha1)

b2sdk.transferer.transferer – Manager of downloaders

class b2sdk.transferer.transferer.**Transferer**(*session*, *account_info*)
Bases: object

Handle complex actions around downloads and uploads to free raw_api from that responsibility.

DEFAULT_MAX_STREAMS = 8

DEFAULT_MIN_PART_SIZE = 104857600

MIN_CHUNK_SIZE = 8192

MAX_CHUNK_SIZE = 1048576

__init__(*session*, *account_info*)

Parameters

- **max_streams** – limit on a number of streams to use when downloading in multiple parts
- **min_part_size** – the smallest part size for which a stream will be run when downloading in multiple parts

download_file_from_url(*url*, *download_dest*, *progress_listener=None*, *range_=None*)

Parameters

- **url** – url from which the file should be downloaded
- **download_dest** – where to put the file when it is downloaded
- **progress_listener** – where to notify about progress downloading
- **range** – 2-element tuple containing data of http Range header

b2sdk.upload_source

class b2sdk.upload_source.**AbstractUploadSource**
Bases: object

The source of data for uploading to b2.

abstract get_content_length()
Return the number of bytes of data in the file.

abstract get_content_sha1()
Return a 40-character string containing the hex SHA1 checksum of the data in the file.

abstract open()
Return a binary file-like object from which the data can be read. :return:

class b2sdk.upload_source.**UploadSourceBytes**(*data_bytes*)
Bases: *b2sdk.upload_source.AbstractUploadSource*

__init__(*data_bytes*)
Initialize self. See help(type(self)) for accurate signature.

get_content_length()
Return the number of bytes of data in the file.

get_content_sha1()
Return a 40-character string containing the hex SHA1 checksum of the data in the file.

open()
Return a binary file-like object from which the data can be read. :return:

class b2sdk.upload_source.**UploadSourceLocalFile**(*local_path*, *content_sha1=None*)
Bases: *b2sdk.upload_source.AbstractUploadSource*

__init__(*local_path*, *content_sha1=None*)
Initialize self. See help(type(self)) for accurate signature.

get_content_length()
Return the number of bytes of data in the file.

get_content_sha1()
Return a 40-character string containing the hex SHA1 checksum of the data in the file.

open()
Return a binary file-like object from which the data can be read. :return:

2.7 Contributors Guide

We encourage outside contributors to perform changes on our codebase. Many such changes have been merged already. In order to make it easier to contribute, core developers of this project:

- provide guidance (through the issue reporting system)
- provide tool assisted code review (through the Pull Request system)
- maintain a set of integration tests (run with a production cloud)
- maintain a set of (well over a hundred) unit tests
- automatically run unit tests on 13 versions of python (including `osx` and `pypy`)
- format the code automatically using yapf
- use static code analysis to find subtle/potential issues with maintainability
- maintain other Continous Integration tools (coverage tracker)

We marked the places in the code which are significantly less intuitive than others in a special way. To find them occurrences, use `git grep '*magic*'`.

To install a development environment, please follow *this link*.

To test in multiple python virtual environments, set the enviroment variable `PYTHON_VIRTUAL_ENVS` to be a space-separated list of their root directories. When set, the makefile will run the unit tests in each of the environments.

Before checking in, use the `pre-commit.sh` script to check code formatting, run unit tests, run integration tests etc.

The integration tests need a file in your home directory called `.b2_auth` that contains two lines with nothing on them but your account ID and application key:

```
accountId
applicationKey
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S