B2_Command_Line_Tool Documentation

Release 3.16.0

Backblaze

Feb 23, 2024

CONTENTS

This program provides command-line access to the B2 service.

There are two flows of authorization:

- call `b2 authorize-account` and have the credentials cached in sqlite
- set `B2_APPLICATION_KEY_ID` and `B2_APPLICATION_KEY` environment variables when running this program

This program caches authentication-related and other data in a local SQLite database. The location of this database is determined in the following way:

If `--profile` arg is provided:

- `XDG_CONFIG_HOME/b2/db-<profile>.sqlite`, if `XDG_CONFIG_HOME` env var is set
- `~/.b2db-{profile}.sqlite`

Otherwise:

- `B2_ACCOUNT_INFO` env var's value, if set
- `~/.b2_account_info`, if it exists
- `XDG_CONFIG_HOME/b2/account_info`, if `XDG_CONFIG_HOME` env var is set
- `~/.b2_account_info`, as default

If the directory `XDG_CONFIG_HOME/b2` does not exist (and is needed), it is created. Please note that the above rules may be changed in next versions of b2sdk, and in order to get reliable authentication file location you should use `b2 get-account-info`.

You can suppress command stdout & stderr output by using `--quiet` option. To supress only progress bar, use `--noProgress` option.

For more details on one command:

```
b2 <command> --help
```

When authorizing with application keys, this tool requires that the key have the `listBuckets` capability so that it can take the bucket names you provide on the command line and translate them into bucket IDs for the B2 Storage service. Each different command may required additional capabilities. You can find the details for each command in the help for that command.

A string provided via an optional environment variable `B2_USER_AGENT_APPEND` will be appended to the User-Agent.

CHAPTER

ONE

DOCUMENTATION INDEX

1.1 Quick Start Guide

1.1.1 Prepare B2 cli

```
$ b2 authorize-account 4ab123456789 001aabbccddeeff12345678901234567890123456789
Using https://api.backblazeb2.com
```

Tip: Get credentials from B2 website

Warning: Local users might be able to access your process list and read command arguments. To avoid exposing credentials, you can provide application key ID and application key using environment variables `B2_APPLICATION_KEY_ID` and `B2_APPLICATION_KEY` respectively. Those will be picked up automatically, so after defining those you'll just need to run `b2 authorize-account` with no extra parameters.

```
$ export B2_APPLICATION_KEY_ID="$(<file-with-key-id.txt)"
$ export B2_APPLICATION_KEY="$(<file-with-key.txt)"
$ b2 authorize-account
Using https://api.backblazeb2.com
```

1.1.2 Synchronization

```
$ b2 sync "/home/user1/b2_example" "b2://bucket1/example-mybucket-b2"
```

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.

1.1.3 Bucket actions

List buckets

```
$ b2 list-buckets
34567890abcdef1234567890  allPublic   example-mybucket-b2-1
34567890123456789abcdef  allPublic   example-mybucket-b2-2
```

Create a bucket

```
$ b2 create_bucket example-mybucket-b2-3 allPublic
...
```

You can optionally store bucket info, CORS rules and lifecycle rules with the bucket.

Delete a bucket

```
$ b2 delete-bucket 'example-mybucket-b2-1'
```

returns 0 if successful, outputs a message and a non-0 return code in case of error.

1.2 Commands

1.2.1 Authorize-account command

Prompts for Backblaze `applicationKeyId` and `applicationKey` (unless they are given on the command line).

You can authorize with either the master application key or a normal application key.

To use the master application key, provide the application key ID and application key from the `B2 Cloud Storage Buckets` page on the web site: https://secure.backblaze.com/b2_buckets.htm

To use a normal application key, created with the `create-key` command or on the web site, provide the application key ID and the application key itself.

You can also optionally provide application key ID and application key using environment variables `B2_APPLICATION_KEY_ID` and `B2_APPLICATION_KEY` respectively.

Stores an account auth token in a local cache, see

```
b2 --help
```

for details on how the location of this cache is determined.

Requires capability:

- **listBuckets**

```
b2 authorize-account [-h]  [applicationKeyId] [applicationKey]
```

Positional Arguments

applicationKeyId

applicationKey

1.2.2 Cancel-all-unfinished-large-files command

Lists all large files that have been started but not finished and cancels them. Any parts that have been uploaded will be deleted.

Requires capability:

- **listFiles**
- **writeFiles**

```
b2 cancel-all-unfinished-large-files [-h] bucketName
```

Positional Arguments

bucketName

1.2.3 Cancel-large-file command

Cancels a large file upload. Used to undo a `start-large-file`.

Cannot be used once the file is finished. After finishing, using `delete-file-version` to delete the large file.

Requires capability:

- **writeFiles**

```
b2 cancel-large-file [-h] fileId
```

Positional Arguments

fileId

1.2.4 Cat command

Download content of a file-like object identified by B2 URI directly to stdout.

If the `tqdm` library is installed, progress bar is displayed on stderr. Without it, simple text progress is printed. Use `--noProgress` to disable progress reporting (marginally improves performance in some cases).

To access SSE-C encrypted files, please set `--sourceServerSideEncryption=SSE-C`. The default algorithm is set to AES256 which can by changed with `--sourceServerSideEncryptionAlgorithm` parameter. Using SSE-C requires providing `B2_SOURCE_SSE_C_KEY_B64` environment variable, containing the base64 encoded encryption key.

Use –write-buffer-size to set the size (in bytes) of the buffer used to write files.

Use –skip-hash-verification to disable hash check on downloaded files.

Requires capability:

- **readFiles**

```
b2 cat [-h] [--noProgress] [--sourceServerSideEncryption {SSE-C}]
       [--sourceServerSideEncryptionAlgorithm {AES256}]
       [--write-buffer-size BYTES] [--skip-hash-verification]
       B2_URI
```

Positional Arguments

B2_URI B2 URI pointing to a file, e.g. b2://yourBucket/file.txt or b2id://fileId

Named Arguments

--noProgress progress will not be reported

Default: False

--sourceServerSideEncryption Possible choices: SSE-C

--sourceServerSideEncryptionAlgorithm Possible choices: AES256

Default: “AES256”

--write-buffer-size

--skip-hash-verification Default: False

1.2.5 Clear-account command

Erases everything in local cache.

See

```
b2 --help
```

for details on how the location of this cache is determined.

```
b2 clear-account [-h]
```

1.2.6 Copy-file-by-id command

Copy a file version to the given bucket (server-side, **not** via download+upload). Copies the contents of the source B2 file to destination bucket and assigns the given name to the new B2 file, possibly setting options like server-side encryption and retention.

> **Warning:** Setting file retention mode to 'compliance' is irreversible - such files can only be ever deleted after their retention period passes, regardless of keys (master or not) used. This is especially dangerous when setting bucket default retention, as it may lead to high storage costs.

By default, it copies the file info and content type, therefore `--contentType` and `--info` are optional. If one of them is set, the other has to be set as well.

To force the destination file to have empty fileInfo, use `--noInfo`.

By default, the whole file gets copied, but you can copy an (inclusive!) range of bytes from the source file to the new file using `--range` option.

Each `--info` entry is in the form `a=b`, you can specify many.

The maximum file size is 5GB or 10TB, depending on capability of installed `b2sdk` version.

To request SSE-B2 or SSE-C encryption for destination files, please set `--destinationServerSideEncryption=SSE-B2/SSE-C`. The default algorithm is set to AES256 which can be changed with `--destinationServerSideEncryptionAlgorithm` parameter. Using SSE-C requires providing `B2_DESTINATION_SSE_C_KEY_B64` environment variable, containing the base64 encoded encryption key. If `B2_DESTINATION_SSE_C_KEY_ID` environment variable is provided, it's value will be saved as `sse_c_key_id` in the uploaded file's fileInfo.

To access SSE-C encrypted files, please set `--sourceServerSideEncryption=SSE-C`. The default algorithm is set to AES256 which can by changed with `--sourceServerSideEncryptionAlgorithm` parameter. Using SSE-C requires providing `B2_SOURCE_SSE_C_KEY_B64` environment variable, containing the base64 encoded encryption key.

Setting file retention settings requires the **writeFileRetentions** capability, and only works in bucket with fileLockEnabled=true. Providing `--fileRetentionMode` requires providing `--retainUntil` which has to be a future timestamp, in the form of an integer representing milliseconds since epoch. Leaving out these options results in a file retained according to bucket defaults.

Setting legal holds requires the **writeFileLegalHolds** capability, and only works in bucket with fileLockEnabled=true.

If either the source or the destination uses SSE-C and `--contentType` and `--info` are not provided, then to perform the copy the source file's metadata has to be fetched first - an additional request to B2 cloud has to be made. To achieve that, provide `--fetchMetadata`. Without that flag, the command will fail.

Requires capability:

- **readFiles** (if `sourceFileId` bucket is private)
- **writeFiles**

```
b2 copy-file-by-id [-h] [--fetchMetadata] [--contentType CONTENTTYPE]
                   [--range RANGE] [--info INFO | --noInfo]
                   [--cache-control CACHE_CONTROL]
                   [--content-disposition CONTENT_DISPOSITION]
                   [--content-encoding CONTENT_ENCODING]
                   [--content-language CONTENT_LANGUAGE] [--expires EXPIRES]
```

(continues on next page)

(continued from previous page)

```
                    [--destinationServerSideEncryption {SSE-B2,SSE-C}]
                    [--destinationServerSideEncryptionAlgorithm {AES256}]
                    [--sourceServerSideEncryption {SSE-C}]
                    [--sourceServerSideEncryptionAlgorithm {AES256}]
                    [--fileRetentionMode {compliance,governance}]
                    [--retainUntil TIMESTAMP] [--legalHold {on,off}]
                    sourceFileId destinationBucketName b2FileName
```

Positional Arguments

sourceFileId

destinationBucketName

b2FileName

Named Arguments

--fetchMetadata Default: False

--contentType

--range

--info

--noInfo Default: False

--cache-control optional Cache-Control header, value based on RFC 2616 section 14.9, example: 'public, max-age=86400')

--content-disposition optional Content-Disposition header, value based on RFC 2616 section 19.5.1, example: 'attachment; filename="fname.ext"'

--content-encoding optional Content-Encoding header, value based on RFC 2616 section 14.11, example: 'gzip'

--content-language optional Content-Language header, value based on RFC 2616 section 14.12, example: 'mi, en'

--expires optional Expires header, value based on RFC 2616 section 14.21, example: 'Thu, 01 Dec 2050 16:00:00 GMT'

--destinationServerSideEncryption Possible choices: SSE-B2, SSE-C

--destinationServerSideEncryptionAlgorithm Possible choices: AES256

Default: "AES256"

--sourceServerSideEncryption Possible choices: SSE-C

--sourceServerSideEncryptionAlgorithm Possible choices: AES256

Default: "AES256"

--fileRetentionMode Possible choices: compliance, governance

--retainUntil

--legalHold Possible choices: on, off

1.2.7 Create-bucket command

Creates a new bucket. Prints the ID of the bucket created.

Optionally stores bucket info, CORS rules and lifecycle rules with the bucket. These can be given as JSON on the command line.

If you want server-side encryption for all of the files that are uploaded to a bucket, you can enable SSE-B2 encryption as a default setting for the bucket. In order to do that pass `--defaultServerSideEncryption=SSE-B2`. The default algorithm is set to AES256 which can by changed with `--defaultServerSideEncryptionAlgorithm` parameter. All uploads to that bucket, from the time default encryption is enabled onward, will then be encrypted with SSE-B2 by default.

To disable default bucket encryption, use `--defaultServerSideEncryption=none`.

If `--defaultServerSideEncryption` is not provided, default server side encryption is determined by the server.

Note: Note that existing files in the bucket are not affected by default bucket encryption settings.

Use *–lifecycleRule* to set lifecycle rule for the bucket. Multiple rules can be specified by repeating the option.

–lifecycleRules option is deprecated and cannot be used together with –lifecycleRule.

Requires capability:

- **writeBuckets**
- **readBucketEncryption**
- **writeBucketEncryption**
- **writeBucketRetentions**

```
b2 create-bucket [-h] [--bucketInfo BUCKETINFO] [--corsRules CORSRULES]
                 [--fileLockEnabled] [--replication REPLICATION]
                 [--defaultServerSideEncryption {SSE-B2,none}]
                 [--defaultServerSideEncryptionAlgorithm {AES256}]
                 [--lifecycleRule LIFECYCLERULES | --lifecycleRules LIFECYCLERULES]
                 bucketName {allPublic,allPrivate}
```

Positional Arguments

bucketName

bucketType Possible choices: allPublic, allPrivate

Named Arguments

--bucketInfo

--corsRules If given, the bucket will have a 'custom' CORS configuration. Accepts a JSON string.

--fileLockEnabled If given, the bucket will have the file lock mechanism enabled. This parameter cannot be changed after bucket creation.

Default: False

--replication

--defaultServerSideEncryption Possible choices: SSE-B2, none

--defaultServerSideEncryptionAlgorithm Possible choices: AES256

Default: "AES256"

--lifecycleRule Lifecycle rule in JSON format. Can be supplied multiple times.

--lifecycleRules (deprecated; use –lifecycleRule instead) List of lifecycle rules in JSON format.

1.2.8 Create-key command

Creates a new application key. Prints the application key information. This is the only time the application key itself will be returned. Listing application keys will show their IDs, but not the secret keys.

The capabilities are passed in as a comma-separated list, like `readFiles,writeFiles`. Optionally, you can pass all capabilities known to this client with `--allCapabilities`.

The `duration` is the length of time (in seconds) the new application key will exist. When the time expires the key will disappear and will no longer be usable. If not specified, the key will not expire.

The `bucket` is the name of a bucket in the account. When specified, the key will only allow access to that bucket.

The `namePrefix` restricts file access to files whose names start with the prefix.

The output is the new application key ID, followed by the application key itself. The two values returned are the two that you pass to `authorize-account` to use the key.

Requires capability:

- **writeKeys**

```
b2 create-key [-h] [--bucket BUCKET] [--namePrefix NAMEPREFIX]
              [--duration DURATION] [--allCapabilities]
              keyName [capabilities]
```

Positional Arguments

keyName

capabilities

Named Arguments

--bucket

--namePrefix

--duration

--allCapabilities Default: False

1.2.9 Delete-bucket command

Deletes the bucket with the given name.

Requires capability:

- **deleteBuckets**

```
b2 delete-bucket [-h] bucketName
```

Positional Arguments

bucketName

1.2.10 Delete-file-version command

Permanently and irrevocably deletes one version of a file.

Specifying the `fileName` is more efficient than leaving it out. If you omit the `fileName`, it requires an initial query to B2 to get the file name, before making the call to delete the file. This extra query requires the `readFiles` capability.

If a file is in governance retention mode, and the retention period has not expired, adding `--bypassGovernance` is required.

Requires capability:

- **deleteFiles**
- **readFiles** (if file name not provided)

and optionally:

- **bypassGovernance**

```
b2 delete-file-version [-h] [--bypassGovernance] [fileName] fileId
```

Positional Arguments

fileName

fileId

Named Arguments

--bypassGovernance Default: False

1.2.11 Delete-key command

Deletes the specified application key by its ID.

Requires capability:

- **deleteKeys**

```
b2 delete-key [-h] applicationKeyId
```

Positional Arguments

applicationKeyId

1.2.12 Download-file command

Downloads the given file-like object, and stores it in the given local file.

If the `tqdm` library is installed, progress bar is displayed on stderr. Without it, simple text progress is printed. Use `--noProgress` to disable progress reporting (marginally improves performance in some cases).

Use –threads to manually adjust number of threads used in the operation. Otherwise, the number of threads will be automatically chosen.

To access SSE-C encrypted files, please set `--sourceServerSideEncryption=SSE-C`. The default algorithm is set to AES256 which can by changed with `--sourceServerSideEncryptionAlgorithm` parameter. Using SSE-C requires providing `B2_SOURCE_SSE_C_KEY_B64` environment variable, containing the base64 encoded encryption key.

Use –write-buffer-size to set the size (in bytes) of the buffer used to write files.

Use –skip-hash-verification to disable hash check on downloaded files.

Use –max-download-streams-per-file to set max num of streams for parallel downloader.

Requires capability:

- **readFiles**

```
b2 download-file [-h] [--threads THREADS]
                 [--max-download-streams-per-file MAX_DOWNLOAD_STREAMS_PER_FILE]
                 [--noProgress] [--sourceServerSideEncryption {SSE-C}]
                 [--sourceServerSideEncryptionAlgorithm {AES256}]
```

(continues on next page)

(continued from previous page)

```
                [--write-buffer-size BYTES] [--skip-hash-verification]
                B2_URI localFileName
```

Positional Arguments

B2_URI B2 URI pointing to a file, e.g. b2://yourBucket/file.txt or b2id://fileId

localFileName

Named Arguments

--threads

--max-download-streams-per-file

--noProgress progress will not be reported

Default: False

--sourceServerSideEncryption Possible choices: SSE-C

--sourceServerSideEncryptionAlgorithm Possible choices: AES256

Default: "AES256"

--write-buffer-size

--skip-hash-verification Default: False

1.2.13 Download-file-by-id command

Downloads the given file-like object, and stores it in the given local file.

If the `tqdm` library is installed, progress bar is displayed on stderr. Without it, simple text progress is printed. Use `--noProgress` to disable progress reporting (marginally improves performance in some cases).

Use –threads to manually adjust number of threads used in the operation. Otherwise, the number of threads will be automatically chosen.

To access SSE-C encrypted files, please set `--sourceServerSideEncryption=SSE-C`. The default algorithm is set to AES256 which can by changed with `--sourceServerSideEncryptionAlgorithm` parameter. Using SSE-C requires providing `B2_SOURCE_SSE_C_KEY_B64` environment variable, containing the base64 encoded encryption key.

Use –write-buffer-size to set the size (in bytes) of the buffer used to write files.

Use –skip-hash-verification to disable hash check on downloaded files.

Use –max-download-streams-per-file to set max num of streams for parallel downloader.

Requires capability:

- **readFiles**

Warning: This command is deprecated. Use `download-file` instead.

```
b2 download-file-by-id [-h] [--threads THREADS]
                       [--max-download-streams-per-file MAX_DOWNLOAD_STREAMS_PER_FILE]
                       [--noProgress] [--sourceServerSideEncryption {SSE-C}]
                       [--sourceServerSideEncryptionAlgorithm {AES256}]
                       [--write-buffer-size BYTES] [--skip-hash-verification]
                       fileId localFileName
```

Positional Arguments

fileId

localFileName

Named Arguments

--threads

--max-download-streams-per-file

--noProgress progress will not be reported

Default: False

--sourceServerSideEncryption Possible choices: SSE-C

--sourceServerSideEncryptionAlgorithm Possible choices: AES256

Default: "AES256"

--write-buffer-size

--skip-hash-verification Default: False

1.2.14 Download-file-by-name command

Downloads the given file-like object, and stores it in the given local file.

If the `tqdm` library is installed, progress bar is displayed on stderr. Without it, simple text progress is printed. Use `--noProgress` to disable progress reporting (marginally improves performance in some cases).

Use –threads to manually adjust number of threads used in the operation. Otherwise, the number of threads will be automatically chosen.

To access SSE-C encrypted files, please set `--sourceServerSideEncryption=SSE-C`. The default algorithm is set to AES256 which can by changed with `--sourceServerSideEncryptionAlgorithm` parameter. Using SSE-C requires providing `B2_SOURCE_SSE_C_KEY_B64` environment variable, containing the base64 encoded encryption key.

Use –write-buffer-size to set the size (in bytes) of the buffer used to write files.

Use –skip-hash-verification to disable hash check on downloaded files.

Use –max-download-streams-per-file to set max num of streams for parallel downloader.

Requires capability:

- **readFiles**

Warning: This command is deprecated. Use `download-file` instead.

```
b2 download-file-by-name [-h] [--threads THREADS]
                         [--max-download-streams-per-file MAX_DOWNLOAD_STREAMS_PER_FILE]
                         [--noProgress] [--sourceServerSideEncryption {SSE-C}]
                         [--sourceServerSideEncryptionAlgorithm {AES256}]
                         [--write-buffer-size BYTES]
                         [--skip-hash-verification]
                         bucketName fileName localFileName
```

Positional Arguments

bucketName

fileName

localFileName

Named Arguments

--threads

--max-download-streams-per-file

--noProgress progress will not be reported

Default: False

--sourceServerSideEncryption Possible choices: SSE-C

--sourceServerSideEncryptionAlgorithm Possible choices: AES256

Default: “AES256”

--write-buffer-size

--skip-hash-verification Default: False

1.2.15 File-info command

Prints all of the information about the object, but not its contents.

Requires capability:

- **readFiles**

```
b2 file-info [-h] B2_URI
```

Positional Arguments

B2_URI B2 URI pointing to a file, e.g. b2://yourBucket/file.txt or b2id://fileId

1.2.16 Get-account-info command

Shows the account ID, key, auth token, URLs, and what capabilities the current application keys has.

```
b2 get-account-info [-h]
```

1.2.17 Get-bucket command

Prints all of the information about the bucket, including bucket info, CORS rules and lifecycle rules.

If `--showSize` is specified, then display the number of files (`fileCount`) in the bucket and the aggregate size of all files (`totalSize`). Hidden files and hide markers are accounted for in the reported number of files, and hidden files also contribute toward the reported aggregate size, whereas hide markers do not. Each version of a file counts as an individual file, and its size contributes toward the aggregate size. Analysis is recursive.

Note: Note that `--showSize` requires multiple API calls, and will therefore incur additional latency, computation, and Class C transactions.

Requires capability:

- **listBuckets**

```
b2 get-bucket [-h] [--showSize] bucketName
```

Positional Arguments

bucketName

Named Arguments

--showSize Default: False

1.2.18 Get-download-auth command

Prints an authorization token that is valid only for downloading files from the given bucket.

The token is valid for the duration specified, which defaults to 86400 seconds (one day).

Only files that match that given prefix can be downloaded with the token. The prefix defaults to "", which matches all files in the bucket.

Requires capability:

- **shareFiles**

```
b2 get-download-auth [-h] [--prefix PREFIX] [--duration DURATION] bucketName
```

Positional Arguments

bucketName

Named Arguments

--prefix Default: “”

--duration Default: 86400

1.2.19 Get-download-url-with-auth command

Prints a URL to download the given file. The URL includes an authorization token that allows downloads from the given bucket for files whose names start with the given file name.

The URL will work for the given file, but is not specific to that file. Files with longer names that start with the give file name can also be downloaded with the same auth token.

The token is valid for the duration specified, which defaults to 86400 seconds (one day).

Requires capability:

- **shareFiles**

```
b2 get-download-url-with-auth [-h] [--duration DURATION] bucketName fileName
```

Positional Arguments

bucketName

fileName

Named Arguments

--duration Default: 86400

1.2.20 Get-file-info command

Prints all of the information about the object, but not its contents.

Requires capability:

- **readFiles**

Warning: This command is deprecated. Use `file-info` instead.

```
b2 get-file-info [-h] fileId
```

Positional Arguments

fileId

1.2.21 Get-url command

Prints an URL that can be used to download the given file, if it is public.

```
b2 get-url [-h] B2_URI
```

Positional Arguments

B2_URI B2 URI pointing to a file, e.g. b2://yourBucket/file.txt or b2id://fileId

1.2.22 Hide-file command

Uploads a new, hidden, version of the given file.

Requires capability:

- **writeFiles**

```
b2 hide-file [-h] bucketName fileName
```

Positional Arguments

bucketName

fileName

1.2.23 install-autocomplete command

Installs autocomplete for supported shells.

Autocomplete is installed for the current user only and will become available after shell reload. Any existing autocomplete configuration for same executable name will be overwritten.

–shell SHELL Shell to install autocomplete for. Autodetected if not specified. Manually specify "bash" to force bash autocomplete installation when running under different shell.

Note: Please note this command WILL modify your shell configuration file (e.g. ~/.bashrc).

```
b2 install-autocomplete [-h] [--shell {bash}]
```

Named Arguments

--shell Possible choices: bash

1.2.24 List-buckets command

Lists all of the buckets in the current account.

Output lines list the bucket ID, bucket type, and bucket name, and look like this:

```
98c960fd1cb4390c5e0f0519  allPublic   my-bucket
```

Alternatively, the `--json` option produces machine-readable output similar (but not identical) to the server api response format.

Requires capability:

- **listBuckets**

```
b2 list-buckets [-h] [--json]
```

Named Arguments

--json Default: False

1.2.25 List-keys command

Lists the application keys for the current account.

The columns in the output are:

- ID of the application key
- Name of the application key
- Name of the bucket the key is restricted to, or - for no restriction
- Date of expiration, or -
- Time of expiration, or -
- File name prefix, in single quotes
- Command-separated list of capabilities

None of the values contain whitespace.

For keys restricted to buckets that do not exist any more, the bucket name is replaced with `id=<bucketId>`, because deleted buckets do not have names any more.

Requires capability:

- **listKeys**

```
b2 list-keys [-h] [--long]
```

Named Arguments

--long Default: False

1.2.26 List-parts command

Lists all of the parts that have been uploaded for the given large file, which must be a file that was started but not finished or canceled.

Requires capability:

- **writeFiles**

```
b2 list-parts [-h] largeFileId
```

Positional Arguments

largeFileId

1.2.27 List-unfinished-large-files command

Lists all of the large files in the bucket that were started, but not finished or canceled.

Requires capability:

- **listFiles**

```
b2 list-unfinished-large-files [-h] bucketName
```

Positional Arguments

bucketName

1.2.28 Ls command

Using the file naming convention that `/` separates folder names from their contents, returns a list of the files and folders in a given folder. If no folder name is given, lists all files at the top level.

The `--long` option produces very wide multi-column output showing the upload date/time, file size, file id, whether it is an uploaded file or the hiding of a file, and the file name. Folders don't really exist in B2, so folders are shown with `-` in each of the fields other than the name.

The `--json` option produces machine-readable output similar to the server api response format.

The `--replication` option adds replication status

The `--versions` option selects all versions of each file, not just the most recent.

The `--recursive` option will descend into folders, and will select only files, not folders.

The `--withWildcard` option will allow using `*`, `?` and `[]` characters in `folderName` as a greedy wildcard, single character wildcard and range of characters. It requires the `--recursive` option. Remember to quote `folderName` to avoid shell expansion.

The –include and –exclude flags can be used to filter the files returned from the server using wildcards. You can specify multiple –include and –exclude filters. The order of filters matters. The *last* matching filter decides whether a file is included or excluded. If the given list of filters contains only INCLUDE filters, then it is assumed that all files are excluded by default.

Examples

Note: Note the use of quotes, to ensure that special characters are not expanded by the shell.

List csv and tsv files (in any directory, in the whole bucket):

```
b2 ls --recursive --withWildcard bucketName "*.[ct]sv"
```

List all info.txt files from buckets bX, where X is any character:

```
b2 ls --recursive --withWildcard bucketName "b?/info.txt"
```

List all pdf files from buckets b0 to b9 (including sub-directories):

```
b2 ls --recursive --withWildcard bucketName "b[0-9]/*.pdf"
```

Requires capability:

- **listFiles**

```
b2 ls [-h] [--long] [--json] [--replication] [--versions] [-r]
      [--withWildcard] [--include FILTERS] [--exclude FILTERS]
      bucketName [folderName]
```

Positional Arguments

bucketName

folderName

Named Arguments

--long	Default: False
--json	Default: False
--replication	Default: False
--versions	Default: False
-r, --recursive	Default: False
--withWildcard	Default: False
--include	Default: []
--exclude	Default: []

1.2.29 Make-friendly-url command

Prints an URL that can be used to download the given file, if it is public.

> **Warning:** This command is deprecated. Use `get-url` instead.

```
b2 make-friendly-url [-h] bucketName fileName
```

Positional Arguments

bucketName

fileName

1.2.30 Make-url command

Prints an URL that can be used to download the given file, if it is public.

> **Warning:** This command is deprecated. Use `get-url` instead.

```
b2 make-url [-h] fileId
```

Positional Arguments

fileId

1.2.31 replication-setup command

Sets up replication between two buckets (potentially from different accounts), creating and replacing keys if necessary.

Requires capabilities on both profiles:

- **listKeys**
- **createKeys**
- **readReplications**
- **writeReplications**

```
b2 replication-setup [-h] [--destination-profile DESTINATION_PROFILE]
                     [--name NAME] [--priority PRIORITY]
                     [--file-name-prefix PREFIX] [--include-existing-files]
                     SOURCE_BUCKET_NAME DESTINATION_BUCKET_NAME
```

Positional Arguments

SOURCE_BUCKET_NAME

DESTINATION_BUCKET_NAME

Named Arguments

--destination-profile

--name name for the new replication rule on the source side

--priority priority for the new replication rule on the source side [1-2147483647]. Will be set automatically when not specified.

--file-name-prefix only replicate files starting with PREFIX

--include-existing-files if given, also replicates files uploaded prior to creation of the replication rule

Default: False

1.2.32 Rm command

Removes a "folder" or a set of files matching a pattern. Use with caution.

Note: `rm` is a high-level command that under the hood utilizes multiple calls to the server, which means the server cannot guarantee consistency between multiple operations. For example if a file matching a pattern is uploaded during a run of `rm` command, it MIGHT be deleted (as "latest") instead of the one present when the `rm` run has started.

In order to safely delete a single file version, please use `delete-file-version`.

To list (but not remove) files to be deleted, use `--dryRun`. You can also list files via `ls` command - the listing behaviour is exactly the same.

Progress is displayed on the console unless `--noProgress` is specified.

Use –threads to manually adjust number of threads used in the operation. Otherwise, the number of threads will be automatically chosen.

The `--versions` option selects all versions of each file, not just the most recent.

The `--recursive` option will descend into folders, and will select only files, not folders.

The `--withWildcard` option will allow using `*`, `?` and `` `[]` `` characters in `folderName` as a greedy wildcard, single character wildcard and range of characters. It requires the `--recursive` option. Remember to quote `folderName` to avoid shell expansion.

The –include and –exclude flags can be used to filter the files returned from the server using wildcards. You can specify multiple –include and –exclude filters. The order of filters matters. The *last* matching filter decides whether a file is included or excluded. If the given list of filters contains only INCLUDE filters, then it is assumed that all files are excluded by default.

The `--dryRun` option prints all the files that would be affected by the command, but removes nothing.

Normally, when an error happens during file removal, log is printed and the command goes further. If any error should be immediately breaking the command, `--failFast` can be passed to ensure that first error will stop the execution. This could be useful to e.g. check whether provided credentials have **deleteFiles** capabilities.

Note: Using `--failFast` doesn't prevent the command from trying to remove further files. It just stops the progress. Since multiple files are removed in parallel, it's possible that just some of them were not reported.

Command returns 0 if all files were removed successfully and a value different from 0 if any file was left.

Examples.

Note: Note the use of quotes, to ensure that special characters are not expanded by the shell.

Note: Use with caution. Running examples presented below can cause data-loss.

Remove all csv and tsv files (in any directory, in the whole bucket):

```
b2 rm --recursive --withWildcard bucketName "*.[ct]sv"
```

Remove all info.txt files from buckets bX, where X is any character:

```
b2 rm --recursive --withWildcard bucketName "b?/info.txt"
```

Remove all pdf files from buckets b0 to b9 (including sub-directories):

```
b2 rm --recursive --withWildcard bucketName "b[0-9]/*.pdf"


Requires capability:

```

(continues on next page)

(continued from previous page)

```
- **listFiles**
- **deleteFiles**
```

```
b2 rm [-h] [--dryRun] [--queueSize QUEUESIZE] [--noProgress] [--failFast]
      [--threads THREADS] [--versions] [-r] [--withWildcard]
      [--include FILTERS] [--exclude FILTERS]
      bucketName [folderName]
```

Positional Arguments

bucketName

folderName

Named Arguments

--dryRun	Default: False
--queueSize	max elements fetched at once for removal, if left unset defaults to twice the number of threads.
--noProgress	Default: False
--failFast	Default: False
--threads	
--versions	Default: False
-r, --recursive	Default: False
--withWildcard	Default: False
--include	Default: []
--exclude	Default: []

1.2.33 Sync command

Copies multiple files from source to destination. Optionally deletes or hides destination files that the source does not have.

The synchronizer can copy files:

- From a B2 bucket to a local destination.
- From a local source to a B2 bucket.
- From one B2 bucket to another.
- Between different folders in the same B2 bucket.

Use `b2://<bucketName>/<prefix>` for B2 paths, e.g. `b2://my-bucket-name/a/path/prefix/`.

Progress is displayed on the console unless `--noProgress` is specified. A list of actions taken is always printed.

Specify `--dryRun` to simulate the actions that would be taken.

To allow sync to run when the source directory is empty, potentially deleting all files in a bucket, specify `--allowEmptySource`. The default is to fail when the specified source directory doesn't exist or is empty. (This check only applies to version 1.0 and later.)

Use –threads to manually adjust number of threads used in the operation. Otherwise, the number of threads will be automatically chosen.

You can alternatively control number of threads per each operation. The number of files processed in parallel is set by `--syncThreads`, the number of files/file parts downloaded in parallel is set by``–downloadThreads``, and the number of files/file parts uploaded in parallel is set by *–uploadThreads* `. All the three parameters can be set to the same value by `--threads`. Experiment with parameters if the defaults are not working well.

Users with low-performance networks may benefit from reducing the number of threads. Using just one thread will minimize the impact on other users of the network.

Note: Note that using multiple threads could be detrimental to the other users on your network.

You can specify `--excludeRegex` to selectively ignore files that match the given pattern. Ignored files will not copy during the sync operation. The pattern is a regular expression that is tested against the full path of each file.

You can specify `--includeRegex` to selectively override ignoring files that match the given `--excludeRegex` pattern by an `--includeRegex` pattern. Similarly to `--excludeRegex`, the pattern is a regular expression that is tested against the full path of each file.

Note: Note that `--includeRegex` cannot be used without `--excludeRegex`.

You can specify `--excludeAllSymlinks` to skip symlinks when syncing from a local source.

When a directory is excluded by using `--excludeDirRegex`, all of the files within it are excluded, even if they match an `--includeRegex` pattern. This means that there is no need to look inside excluded directories, and you can exclude directories containing files for which you don't have read permission and avoid getting errors.

The `--excludeDirRegex` is a regular expression that is tested against the full path of each directory. The path being matched does not have a trailing `/`, so don't include on in your regular expression.

Multiple regex rules can be applied by supplying them as pipe delimited instructions. Note that the regex for this command is Python regex. Reference: https://docs.python.org/3/library/re.html

Regular expressions are considered a match if they match a substring starting at the first character. `.*e` will match `hello`. This is not ideal, but we will maintain this behavior for compatibility. If you want to match the entire path, put a `$` at the end of the regex, such as `.*llo$`.

You can specify `--excludeIfModifiedAfter` to selectively ignore file versions (including hide markers) which were synced after given time (for local source) or ignore only specific file versions (for b2 source). Ignored files or file versions will not be taken for consideration during sync. The time should be given as a seconds timestamp (e.g. "1367900664") If you need milliseconds precision, put it after the comma (e.g. "1367900664.152")

Files are considered to be the same if they have the same name and modification time. This behaviour can be changed using the `--compareVersions` option. Possible values are:

- `none`: Comparison using the file name only
- `modTime`: Comparison using the modification time (default)
- `size`: Comparison using the file size

A future enhancement may add the ability to compare the SHA1 checksum of the files.

Fuzzy comparison of files based on modTime or size can be enabled by specifying the `--compareThreshold` option. This will treat modTimes (in milliseconds) or sizes (in bytes) as the same if they are within the comparison threshold. Files that match, within the threshold, will not be synced. Specifying `--verbose` and `--dryRun` can be useful to determine comparison value differences.

When a destination file is present that is not in the source, the default is to leave it there. Specifying `--delete` means to delete destination files that are not in the source.

When the destination is B2, you have the option of leaving older versions in place. Specifying `--keepDays` will delete any older versions more than the given number of days old, based on the modification time of the file. This option is not available when the destination is a local folder.

Files at the source that have a newer modification time are always copied to the destination. If the destination file is newer, the default is to report an error and stop. But with `--skipNewer` set, those files will just be skipped. With `--replaceNewer` set, the old file from the source will replace the newer one in the destination.

To make the destination exactly match the source, use:

```
b2 sync --delete --replaceNewer ... ...
```

Warning: Using `--delete` deletes files! We recommend not using it. If you use `--keepDays` instead, you will have some time to recover your files if you discover they are missing on the source end.

To make the destination match the source, but retain previous versions for 30 days:

```
b2 sync --keepDays 30 --replaceNewer ... b2://...
```

Example of sync being used with `--excludeRegex`. This will ignore `.DS_Store` files and `.Spotlight-V100` folders:

```
b2 sync --excludeRegex '(.*\.DS_Store)|(.*\.Spotlight-V100)' ... b2://...
```

To request SSE-B2 or SSE-C encryption for destination files, please set `--destinationServerSideEncryption=SSE-B2/SSE-C`. The default algorithm is set to AES256 which can be changed with `--destinationServerSideEncryptionAlgorithm` parameter. Using SSE-C requires providing `B2_DESTINATION_SSE_C_KEY_B64` environment variable, containing the base64 encoded encryption key. If `B2_DESTINATION_SSE_C_KEY_ID` environment variable is provided, it's value will be saved as `sse_c_key_id` in the uploaded file's fileInfo.

To access SSE-C encrypted files, please set `--sourceServerSideEncryption=SSE-C`. The default algorithm is set to AES256 which can by changed with `--sourceServerSideEncryptionAlgorithm` parameter. Using SSE-C requires providing `B2_SOURCE_SSE_C_KEY_B64` environment variable, containing the base64 encoded encryption key.

Use –write-buffer-size to set the size (in bytes) of the buffer used to write files.

Use –skip-hash-verification to disable hash check on downloaded files.

Use –max-download-streams-per-file to set max num of streams for parallel downloader.

Use –incrementalMode to allow for incremental file uploads to safe bandwidth. This will only affect files, which have been appended to since last upload.

Requires capabilities:

- **listFiles**
- **readFiles** (for downloading)
- **writeFiles** (for uploading)

```
b2 sync [-h] [--noProgress] [--dryRun] [--allowEmptySource]
        [--excludeAllSymlinks] [--syncThreads SYNCTHREADS]
        [--downloadThreads DOWNLOADTHREADS] [--uploadThreads UPLOADTHREADS]
        [--compareVersions {none,modTime,size}] [--compareThreshold MILLIS]
        [--excludeRegex REGEX] [--includeRegex REGEX]
        [--excludeDirRegex REGEX] [--excludeIfModifiedAfter TIMESTAMP]
        [--threads THREADS] [--destinationServerSideEncryption {SSE-B2,SSE-C}]
        [--destinationServerSideEncryptionAlgorithm {AES256}]
        [--sourceServerSideEncryption {SSE-C}]
        [--sourceServerSideEncryptionAlgorithm {AES256}]
        [--write-buffer-size BYTES] [--skip-hash-verification]
        [--max-download-streams-per-file MAX_DOWNLOAD_STREAMS_PER_FILE]
        [--incrementalMode] [--skipNewer | --replaceNewer]
        [--delete | --keepDays DAYS]
        source destination
```

Positional Arguments

source

destination

Named Arguments

--noProgress Default: False

--dryRun Default: False

--allowEmptySource Default: False

--excludeAllSymlinks Default: False

--syncThreads Default: 10

--downloadThreads Default: 10

--uploadThreads Default: 10

--compareVersions Possible choices: none, modTime, size

Default: “modTime”

--compareThreshold

--excludeRegex Default: []

--includeRegex Default: []

--excludeDirRegex Default: []

--excludeIfModifiedAfter

--threads

--destinationServerSideEncryption Possible choices: SSE-B2, SSE-C

--destinationServerSideEncryptionAlgorithm Possible choices: AES256

Default: “AES256”

--sourceServerSideEncryption Possible choices: SSE-C

--sourceServerSideEncryptionAlgorithm Possible choices: AES256

Default: "AES256"

--write-buffer-size

--skip-hash-verification Default: False

--max-download-streams-per-file

--incrementalMode Default: False

--skipNewer Default: False

--replaceNewer Default: False

--delete Default: False

--keepDays

1.2.34 Update-bucket command

Updates the `bucketType` of an existing bucket. Prints the ID of the bucket updated.

Optionally stores bucket info, CORS rules and lifecycle rules with the bucket. These can be given as JSON on the command line.

If you want server-side encryption for all of the files that are uploaded to a bucket, you can enable SSE-B2 encryption as a default setting for the bucket. In order to do that pass `--defaultServerSideEncryption=SSE-B2`. The default algorithm is set to AES256 which can by changed with `--defaultServerSideEncryptionAlgorithm` parameter. All uploads to that bucket, from the time default encryption is enabled onward, will then be encrypted with SSE-B2 by default.

To disable default bucket encryption, use `--defaultServerSideEncryption=none`.

If `--defaultServerSideEncryption` is not provided, default server side encryption is determined by the server.

Note: Note that existing files in the bucket are not affected by default bucket encryption settings.

Use *–lifecycleRule* to set lifecycle rule for the bucket. Multiple rules can be specified by repeating the option.

–lifecycleRules option is deprecated and cannot be used together with –lifecycleRule.

To set a default retention for files in the bucket `--defaultRetentionMode` and `--defaultRetentionPeriod` have to be specified. The latter one is of the form "X days|years".

Warning: Setting file retention mode to 'compliance' is irreversible - such files can only be ever deleted after their retention period passes, regardless of keys (master or not) used. This is especially dangerous when setting bucket default retention, as it may lead to high storage costs.

This command can be used to set the bucket's `fileLockEnabled` flag to `true` using the `--fileLockEnabled` option. This can only be done if the bucket is not set up as a replication source.

Warning: Once `fileLockEnabled` is set, it can NOT be reverted back to `false`

Please note that replication from file-lock-enabled bucket to file-lock-disabled bucket is not allowed, therefore if file lock is enabled on a bucket, it can never again be the replication source bucket for a file-lock-disabled destination.

Additionally in a file-lock-enabled bucket the file metadata limit will be decreased from 7000 bytes to 2048 bytes for new file versions Please consult `b2_update_bucket` official documentation for further guidance.

Requires capability:

- **writeBuckets**
- **readBucketEncryption**

and for some operations:

- **writeBucketRetentions**
- **writeBucketEncryption**

```
b2 update-bucket [-h] [--bucketInfo BUCKETINFO] [--corsRules CORSRULES]
                 [--defaultRetentionMode {compliance,governance,none}]
                 [--defaultRetentionPeriod period] [--replication REPLICATION]
                 [--fileLockEnabled]
                 [--defaultServerSideEncryption {SSE-B2,none}]
                 [--defaultServerSideEncryptionAlgorithm {AES256}]
                 [--lifecycleRule LIFECYCLERULES | --lifecycleRules LIFECYCLERULES]
                 bucketName [{allPublic,allPrivate}]
```

Positional Arguments

bucketName

bucketType Possible choices: allPublic, allPrivate

Named Arguments

--bucketInfo

--corsRules If given, the bucket will have a 'custom' CORS configuration. Accepts a JSON string.

--defaultRetentionMode Possible choices: compliance, governance, none

--defaultRetentionPeriod

--replication

--fileLockEnabled If given, the bucket will have the file lock mechanism enabled. This parameter cannot be changed back.

--defaultServerSideEncryption Possible choices: SSE-B2, none

--defaultServerSideEncryptionAlgorithm Possible choices: AES256

Default: "AES256"

--lifecycleRule Lifecycle rule in JSON format. Can be supplied multiple times.

--lifecycleRules (deprecated; use –lifecycleRule instead) List of lifecycle rules in JSON format.

1.2.35 Update-file-legal-hold command

Only works in buckets with fileLockEnabled=true.

Specifying the `fileName` is more efficient than leaving it out. If you omit the `fileName`, it requires an initial query to B2 to get the file name, before making the call to delete the file. This extra query requires the `readFiles` capability.

Requires capability:

- **writeFileLegalHolds**
- **readFiles** (if file name not provided)

```
b2 update-file-legal-hold [-h] [fileName] fileId {on,off}
```

Positional Arguments

fileName

fileId

legalHold Possible choices: on, off

1.2.36 Update-file-retention command

Only works in buckets with fileLockEnabled=true. Providing a `retentionMode` other than `none` requires providing `retainUntil`, which has to be a future timestamp in the form of an integer representing milliseconds since epoch.

If a file already is in governance mode, disabling retention or shortening it's period requires providing `--bypassGovernance`.

If a file already is in compliance mode, disabling retention or shortening it's period is impossible.

Warning: Setting file retention mode to 'compliance' is irreversible - such files can only be ever deleted after their retention period passes, regardless of keys (master or not) used. This is especially dangerous when setting bucket default retention, as it may lead to high storage costs.

In both cases prolonging the retention period is possible. Changing from governance to compliance is also supported.

Specifying the `fileName` is more efficient than leaving it out. If you omit the `fileName`, it requires an initial query to B2 to get the file name, before making the call to delete the file. This extra query requires the `readFiles` capability.

Requires capability:

- **writeFileRetentions**
- **readFiles** (if file name not provided)

and optionally:

- **bypassGovernance**

```
b2 update-file-retention [-h] [--retainUntil TIMESTAMP] [--bypassGovernance]
                         [fileName] fileId {governance,compliance,none}
```

Positional Arguments

fileName

fileId

retentionMode Possible choices: governance, compliance, none

Named Arguments

--retainUntil

--bypassGovernance Default: False

1.2.37 Upload-file command

Uploads one file to the given bucket.

Uploads the contents of the local file, and assigns the given name to the B2 file, possibly setting options like server-side encryption and retention.

A FIFO file (such as named pipe) can be given instead of regular file.

By default, upload_file will compute the sha1 checksum of the file to be uploaded. But, if you already have it, you can provide it on the command line to save a little time.

Warning: Setting file retention mode to 'compliance' is irreversible - such files can only be ever deleted after their retention period passes, regardless of keys (master or not) used. This is especially dangerous when setting bucket default retention, as it may lead to high storage costs.

Content type is optional. If not set, it will be guessed.

The maximum number of upload threads to use to upload parts of a large file is specified by `--threads`. It has no effect on "small" files (under 200MB as of writing this). Default is 10.

Each fileInfo is of the form `a=b`.

By default, the file is broken into many parts to maximize upload parallelism and increase speed. Setting `--minPartSize` controls the minimal upload file part size. Part size must be in 5MB to 5GB range. Reference: https://www.backblaze.com/docs/cloud-storage-create-large-files-with-the-native-api

If the `tqdm` library is installed, progress bar is displayed on stderr. Without it, simple text progress is printed. Use `--noProgress` to disable progress reporting (marginally improves performance in some cases).

Use –threads to manually adjust number of threads used in the operation. Otherwise, the number of threads will be automatically chosen.

To request SSE-B2 or SSE-C encryption for destination files, please set `--destinationServerSideEncryption=SSE-B2/SSE-C`. The default algorithm is set to AES256 which can be changed with `--destinationServerSideEncryptionAlgorithm` parameter. Using SSE-C requires providing `B2_DESTINATION_SSE_C_KEY_B64` environment variable, containing the base64 encoded encryption key. If `B2_DESTINATION_SSE_C_KEY_ID` environment variable is provided, it's value will be saved as `sse_c_key_id` in the uploaded file's fileInfo.

Setting file retention settings requires the **writeFileRetentions** capability, and only works in bucket with fileLockEnabled=true. Providing `--fileRetentionMode` requires providing `--retainUntil` which has to be a future timestamp, in the form of an integer representing milliseconds since epoch. Leaving out these options results in a file retained according to bucket defaults.

Setting legal holds requires the **writeFileLegalHolds** capability, and only works in bucket with fileLockEnabled=true.

Use –incrementalMode to allow for incremental file uploads to safe bandwidth. This will only affect files, which have been appended to since last upload.

The `--custom-upload-timestamp`, in milliseconds-since-epoch, can be used to artificially change the upload timestamp of the file for the purpose of preserving retention policies after migration of data from other storage. The access to this feature is restricted - if you really need it, you'll need to contact customer support to enable it temporarily for your account.

Requires capability:

- **writeFiles**

```
b2 upload-file [-h] [--contentType CONTENTTYPE] [--sha1 SHA1] [--info INFO]
               [--custom-upload-timestamp CUSTOM_UPLOAD_TIMESTAMP]
               [--cache-control CACHE_CONTROL]
               [--content-disposition CONTENT_DISPOSITION]
               [--content-encoding CONTENT_ENCODING]
               [--content-language CONTENT_LANGUAGE] [--expires EXPIRES]
               [--minPartSize MINPARTSIZE] [--threads THREADS] [--noProgress]
               [--destinationServerSideEncryption {SSE-B2,SSE-C}]
               [--destinationServerSideEncryptionAlgorithm {AES256}]
               [--legalHold {on,off}]
               [--fileRetentionMode {compliance,governance}]
               [--retainUntil TIMESTAMP] [--incrementalMode]
               bucketName localFilePath b2FileName
```

Positional Arguments

bucketName name of the bucket where the file will be stored

localFilePath path of the local file or stream to be uploaded

b2FileName name file will be given when stored in B2

Named Arguments

--contentType MIME type of the file being uploaded. If not set it will be guessed.

--sha1 SHA-1 of the data being uploaded for verifying file integrity

--info additional file info to be stored with the file. Can be used multiple times for different information.

Default: []

--custom-upload-timestamp overrides object creation date. Expressed as a number of milliseconds since epoch.

--cache-control optional Cache-Control header, value based on RFC 2616 section 14.9, example: 'public, max-age=86400')

--content-disposition optional Content-Disposition header, value based on RFC 2616 section 19.5.1, example: 'attachment; filename="fname.ext"'

--content-encoding optional Content-Encoding header, value based on RFC 2616 section 14.11, example: 'gzip'

--content-language optional Content-Language header, value based on RFC 2616 section 14.12, example: 'mi, en'

--expires optional Expires header, value based on RFC 2616 section 14.21, example: 'Thu, 01 Dec 2050 16:00:00 GMT'

--minPartSize minimum part size in bytes

--threads

--noProgress progress will not be reported

Default: False

--destinationServerSideEncryption Possible choices: SSE-B2, SSE-C

--destinationServerSideEncryptionAlgorithm Possible choices: AES256

Default: "AES256"

--legalHold Possible choices: on, off

--fileRetentionMode Possible choices: compliance, governance

--retainUntil

--incrementalMode Default: False

1.2.38 Version command

Prints the version number of this tool.

```
b2 version [-h] [--short]
```

Named Arguments

--short Default: False

1.3 Replication

If you have access to accounts hosting both source and destination bucket (it can be the same account), we recommend using `replication-setup` command described below. Otherwise use *manual setup*.

1.3.1 Automatic setup

Setup replication

```
$ b2 replication-setup --destination-profile myprofile2 my-bucket my-bucket2
```

You can optionally choose source rule priority and source rule name. See *replication-setup command*.

Note: `replication-setup` will reuse or provision a source key with no prefix and full reading capabilities and a destination key with no prefix and full writing capabilities

---

1.3.2 Manual setup

Setup source key

```
$ b2 create-key my-bucket-rplsrc readFiles,readFileLegalHolds,readFileRetentions
0014ab123456789000000000123 K001ZA12345678901234567890ABCDE
```

Setup source replication

```
$ b2 update-bucket --replication '{
    "asReplicationSource": {
        "replicationRules": [
            {
                "destinationBucketId": "85644d98debc657d880b0e1e",
                "fileNamePrefix": "files-to-share/",
                "includeExistingFiles": false,
                "isEnabled": true,
                "priority": 128,
                "replicationRuleName": "my-replication-rule-name"
            }
        ],
        "sourceApplicationKeyId": "0014ab123456789000000000123"
    }
}' my-bucket
```

Setup destination key

```
$ b2 create-key --profile myprofile2 my-bucket-rpldst writeFiles,writeFileLegalHolds,
↪writeFileRetentions,deleteFiles
0024ab2345678900000000234 K001YYABCDE12345678901234567890
```

Setup destination replication

```
$ b2 update-bucket --profile myprofile2 --replication '{
    "asReplicationDestination": {
        "sourceToDestinationKeyMapping": {
            "0014ab1234567890000000123": "0024ab234567890000000234"
        }
    }
}' my-bucket
```

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search