B2_Command_Line_Tool Documentation

Release 2.1.0

Backblaze

Jan 06, 2021

CONTENTS:

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 ~/.b2_account_info by default, or the file specified by the B2_ACCOUNT_INFO environment variable.

Requires capability: listBuckets

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

1.1 Positional Arguments

applicationKeyId

applicationKey

TWO

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
```

2.1 Positional Arguments

bucketName

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
```

3.1 Positional Arguments

fileId

FOUR

CLEAR-ACCOUNT COMMAND

Erases everything in ~/.b2_account_info. Location of file can be overridden by setting B2_ACCOUNT_INFO.

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

FIVE

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.

By default, it copies the file info and content type. You can replace those by setting the metadataDirective to "replace".

—contentType and –info should only be provided when –metadataDirective is set to "replace" and should not be provided when –metadataDirective is set to "copy".

–contentType and –info are optional. If not set, they will be set based on the source file.

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.

Requires capability: readFiles (if sourceFileId bucket is private) and writeFiles

```
b2 copy-file-by-id [-h] [--metadataDirective {copy,replace}]
                   [--contentType CONTENTTYPE] [--range RANGE] [--info INFO]
                   sourceFileId destinationBucketName b2FileName
```

5.1 Positional Arguments

sourceFileId

destinationBucketName

b2FileName

5.2 Named Arguments

--metadataDirective   Possible choices: copy, replace

--contentType

--range

--info   Default: []

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.

Requires capability: writeBuckets

```
b2 create-bucket [-h] [--bucketInfo BUCKETINFO] [--corsRules CORSRULES]
                 [--lifecycleRules LIFECYCLERULES]
                 bucketName bucketType
```

6.1 Positional Arguments

bucketName

bucketType

6.2 Named Arguments

--bucketInfo

--corsRules

--lifecycleRules

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".

The 'duration' is the length of time 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]
              keyName capabilities
```

7.1 Positional Arguments

keyName

capabilities

7.2 Named Arguments

--bucket

--namePrefix

--duration

EIGHT

DELETE-BUCKET COMMAND

Deletes the bucket with the given name.

Requires capability: deleteBuckets

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

8.1 Positional Arguments

bucketName

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.

Requires capability: deleteFiles, readFiles (if file name not provided)

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

9.1 Positional Arguments

fileName

fileId

DELETE-KEY COMMAND

Deletes the specified application key by its 'ID'.

Requires capability: deleteKeys

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

10.1 Positional Arguments

applicationKeyId

ELEVEN

DOWNLOAD-FILE-BY-ID COMMAND

Downloads the given file, 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.

Requires capability: readFiles

```
b2 download-file-by-id [-h] [--noProgress] fileId localFileName
```

11.1 Positional Arguments

fileId

localFileName

11.2 Named Arguments

--noProgress   Default: False

DOWNLOAD-FILE-BY-NAME COMMAND

Downloads the given file, 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.

Requires capability: readFiles

```
b2 download-file-by-name [-h] [--noProgress]
                         bucketName b2FileName localFileName
```

12.1 Positional Arguments

bucketName

b2FileName

localFileName

12.2 Named Arguments

--noProgress   Default: False

THIRTEEN

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]
```

FOURTEEN

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 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
```

14.1 Positional Arguments

bucketName

14.2 Named Arguments

--showSize   Default: False

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
```

15.1 Positional Arguments

bucketName

15.2 Named Arguments

--prefix   Default: ""

--duration   Default: 86400

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
```

16.1 Positional Arguments

bucketName

fileName

16.2 Named Arguments

--duration   Default: 86400

SEVENTEEN

GET-FILE-INFO COMMAND

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

Requires capability: readFiles

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

17.1 Positional Arguments

fileId

HIDE-FILE COMMAND

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

Requires capability: writeFiles

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

18.1 Positional Arguments

bucketName

fileName

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]
```

19.1 Named Arguments

--json   Default: False

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]
```

20.1 Named Arguments

--long   Default: False

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
```

21.1 Positional Arguments

largeFileId

TWENTYTWO

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
```

22.1 Positional Arguments

bucketName

TWENTYTHREE

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 –versions option shows all versions of each file, not just the most recent.

The –recursive option will descend into folders, and will show only files, not folders.

Requires capability: listFiles

```
b2 ls [-h] [--long] [--json] [--versions] [--recursive]
      bucketName [folderName]
```

23.1 Positional Arguments

bucketName

folderName

23.2 Named Arguments

--long	Default: False
--json	Default: False
--versions	Default: False
--recursive	Default: False

TWENTYFOUR

MAKE-FRIENDLY-URL COMMAND

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

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

24.1 Positional Arguments

bucketName

fileName

MAKE-URL COMMAND

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

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

25.1 Positional Arguments

fileId

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.)

Users with high-performance networks, or file sets with very small files, will benefit from multi-threaded uploads. The default number of threads is 10. Experiment with the –threads parameter if the default is 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 that using multiple threads will usually 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 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/2/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: sphinx-build 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: sphinx-build sync –keepDays 30 –replaceNewer … b2://…

Example of sync being used with excludeRegex. This will ignore .DS_Store files and .Spotlight-V100 folders sphinx-build sync -excludeRegex '(.*.DS_Store)|(.*.Spotlight-V100)' … b2://…

Requires capabilities: listFiles, readFiles (for downloading), writeFiles (for uploading)

```
b2 sync [-h] [--noProgress] [--dryRun] [--allowEmptySource]
        [--excludeAllSymlinks] [--threads THREADS]
        [--compareVersions {none,modTime,size}] [--compareThreshold MILLIS]
        [--excludeRegex REGEX] [--includeRegex REGEX]
        [--excludeDirRegex REGEX] [--excludeIfModifiedAfter TIMESTAMP]
        [--skipNewer | --replaceNewer] [--delete | --keepDays DAYS]
        source destination
```

26.1 Positional Arguments

source

destination

26.2 Named Arguments

--noProgress	Default: False
--dryRun	Default: False
--allowEmptySource	Default: False
--excludeAllSymlinks	Default: False
--threads	Default: 10
--compareVersions	Possible choices: none, modTime, size
	Default: "modTime"
--compareThreshold	
--excludeRegex	Default: []
--includeRegex	Default: []
--excludeDirRegex	Default: []
--excludeIfModifiedAfter	
--skipNewer	Default: False
--replaceNewer	Default: False
--delete	Default: False
--keepDays	

TWENTYSEVEN

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.

Requires capability: writeBuckets

```
b2 update-bucket [-h] [--bucketInfo BUCKETINFO] [--corsRules CORSRULES]
                 [--lifecycleRules LIFECYCLERULES]
                 bucketName bucketType
```

27.1 Positional Arguments

bucketName

bucketType

27.2 Named Arguments

--bucketInfo

--corsRules

--lifecycleRules

TWENTYEIGHT

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.

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.

Content type is optional. If not set, it will be set based on the file extension.

By default, the file is broken into as many parts as possible to maximize upload parallelism and increase speed. The minimum that B2 allows is 100MB. Setting –minPartSize to a larger value will reduce the number of parts uploaded when uploading a large file.

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). Default is 10.

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.

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

Requires capability: writeFiles

```
b2 upload-file [-h] [--noProgress] [--quiet] [--contentType CONTENTTYPE]
               [--minPartSize MINPARTSIZE] [--sha1 SHA1] [--threads THREADS]
               [--info INFO]
               bucketName localFilePath b2FileName
```

28.1 Positional Arguments

bucketName

localFilePath

b2FileName

28.2 Named Arguments

--noProgress	Default: False
--quiet	Default: False
--contentType	
--minPartSize	
--sha1	
--threads	Default: 10
--info	Default: []

TWENTYNINE

VERSION COMMAND

Prints the version number of this tool.

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

THIRTY

INDICES AND TABLES

- genindex
- modindex
- search