B2 Command-Line Tool (API behavior)

This will cover documentation on calls made in the B2 Command-Line Tool (CLI) beyond the list of commands in the installation articles. We will be referencing the output from the `b2 help` call in the CLI rather than the B2 docs and will go into a bit more detail than the individual help guides for each call.

Note: The calls in the CLI will be a bit different than their typical counterparts as listed in the B2 docs. Aside from different mechanics and behavior, the name of the call will be entered into the CLI a little differently than the name in the B2 docs. There is no "-" or "_' after "b2".

⚠ *Below, the individual arguments will have optional portions of the call which will be in [blue] and **required portions of the call**.

Required portions of the call will be position and order specific.

All calls will have [-h] which brings up small guide on how to use the command in the CLI.

All calls can include the optional --verbose which will display the step by step process by the API. If there is an error or issue along the way, it will be displayed with the verbose argument.

b2 authorize-account

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

Required information:

- keyID and application key
  - Can be either Master or created key.

Notes:

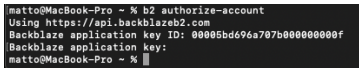
There are two ways to call this command:

1. Type in the call with all of the arguments and press enter to authenticate.
2. Type in the call without any of the arguments and press enter. The CLI will ask you to input each argument individually.
   a. When inputting the applicationKey, it will not be displayed in the terminal or command window.
   b. This method is useful for if you are sharing your screen for others and you do not want to them see your applicationKey.

Output:
When successful, there will be no output.

If it fails, it will tell you "*ERROR: unable to authorize account: Invalid authorization token. Server said: (bad_auth_token)*".

Success:



Link to screenshot

b2 cancel-all-unfinished-large-files

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

Required information:

- `bucketName` - Name of the bucket whose unfinished large files will be canceled.

Notes:

This will list the fileId for all unfinished large files and then will cancel and delete all parts of them.

Output:

Input	Output

b2 cancel-all-unfinished-large-files `mattttata`	`matto@MacBook-Pro ~ % b2 cancel-all-unfinished-large-files mattttata`
	`4_zd0c5ab2db6c916ea7780071b_f223a22a97edcbdc0_d20210414_m221323_c000_v0001066_t0003 canceled`
	`4_zd0c5ab2db6c916ea7780071b_f20910d510ea1ec58_d20210414_m235921_c000_v0001078_t0010 canceled`
	`4_zd0c5ab2db6c916ea7780071b_f203a4f71b70fc33b_d20210414_m235921_c000_v0001079_t0042 canceled`
	`4_zd0c5ab2db6c916ea7780071b_f20910d510ea1ec59_d20210414_m235921_c000_v0001081_t0050 canceled`
	`4_zd0c5ab2db6c916ea7780071b_f2138ccfa96b34b1f_d20210420_m210850_c000_v0001081_t0023 canceled`
	`matto@MacBook-Pro ~ %`
	Link to screenshot

b2 cancel-large-file

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

Require information:

- `fileId` - The fileId of the unfinished large file that will be canceled.
  - This can be found with `b2 list-unfinished-large-files`.

Notes:

This will list the fileId for the unfinished large file and then will cancel and delete all parts of it.

Input	Output

b2 cancel-large-file `4_z4015dbad2609f66a7780071b_f201b37dab303575e_d20210426_m191704_c000_v0001158_t0041`	`matto@MacBook-Pro ~ % b2 cancel-large-file 4_z4015dbad2609f66a7780071b_f201b37dab303575e_d20210426_m191704_c000_v0001158_t0041`
	`4_z4015dbad2609f66a7780071b_f201b37dab303575e_d20210426_m191704_c000_v0001158_t0041 canceled`
	Link to screenshot

b2 clear-account

b2 clear-account [-h]

No required information.

Notes:

This will just invalidate the current session's authorization token. This is essentially logging out of the b2 cli.

Output:

There will be no response from the cli.

b2 copy-file-by-id

b2 copy-file-by-id [-h] [--metadataDirective {copy,replace}] [--contentType CONTENTTYPE] [--range RANGE] [--info INFO] [--destinationServerSideEncryption {SSE-B2}] [--destinationServerSideEncryptionAlgorithm {AES256}] `sourceFileId destinationBucketName b2FileName`

Required information:

- `sourceFileId` - File ID for the file stored in bucket.
  - This can be found with `b2 ls` with the optional [--long] argument.
- `destinationBucketName` - Name of destination bucket in the same account.
- `b2FileName` - User chosen file name as it will be stored in B2.
  - Does not have to match the original.

Notes:

This call can be used for a file in one bucket to create a copy in the same or another bucket in the same account. It cannot be used to move a file from one account to another.

The copy process happen server-side. No downloading and reuploading necessary.

When entering the `b2FileName`, a prefix can be added as a folder path. If the folder path does not exist yet, it will be created when the call is sent.

Optional arguments:

The option [--metadataDirective {copy,replace}] will allow you to copy or replace the metadata

- By default, it copies the file info and content-type. You can replace those by setting the [--metadataDirective {copy,replace}] to replace.
  - For this call, [--contentType CONTENTTYPE] and [--info INFO] should only be provided (and MUST be provided) if [--metadataDirective {copy,replace}] is set to replace.
  - [--contentType CONTENTTYPE] - This is to set the content-type to something other than the original metadata. If the content-type is to stay the same, it should still be stated in the call. Here is a list of possible content-types: https://www.backblaze.com/b2/docs/content-types.html
  - [--info INFO] - These are variables and values chosen and defined by the user.
    - This is to be formatted as `VARIABLE=VALUE`. Example below.
    - If more than one `VARIABLE=VALUE` is to be specified, and additional --info entry must be made. Example below.
- If left out, this choice will just default to copy.

The option [--range RANGE] allows you to specify a part of a file to be copied.

- RANGE should be a pair of numbers (number of bytes) separated by a comma.
- The "beginning of the file" would be 0, so if you want to copy the first half of the file of a 100,000 byte file, you would need to set --range 0,50000
  - It will error out if the first number is larger than the second.
  - It does not have to start at the beginning of the file. It can be something like 10,51000

The option [--destinationServerSideEncryption {SSE-B2}] will set Server-Side Encryption for the file at the file level.

- This does not require Default Server-Side Encryption enabled on the bucket.
- This will default to setting [--destinationServerSideEncryptionAlgorithm {AES256}] to AES256, so this is not a required argument to include when setting --destinationServerSideEncryption SSE-B2.

Output:

The call will give back information similar to `b2 get-file-info`.

Input	Output

b2 copy-file-by-id `4_zd0c5ab2db6c916ea7780071b_f103f3baad8578773_d20210417_m200400_c000_v0001034_t0021` `matttt ata newfolder/bird.jpeg`	 Link to screenshot
b2 copy-file-by-id --metadataDirective replace --contentType video /mp4 --info fileOwner=matt `4_z327a4f1ee3e42c586bcd0619_f11811702e1d33ee9_d20191231_m172906_c000_v0001064_t0027` `himatt 12.mp4` For multiple --info entries, the call will look like this: b2 copy-file-by-id --metadataDirective replace --contentType video /mp4 --info hello=matt --info matt=hello`4_z327a4f1ee3e42c586bcd0619_f11811702e1d33ee9_d20191231_m172906_c000_v0001064_t0027` `himatt 12.mp4`	 Link to screenshot
b2 copy-file-by-id --range 0,50 `4_z327a4f1ee3e42c586bcd0619_f11811702e1d33ee9_d20191231_m172906_c000_v0001064_t0027` `himatt 12.mp4`	 Link to screenshot

b2 create-bucket

b2 create-bucket [-h] [--bucketInfo BUCKETINFO] [--corsRules CORSRULES] [--lifecycleRules LIFECYCLERULES] [--defaultServerSideEncryption {SSE-B2,none}] [--defaultServerSideEncryptionAlgorithm {AES256}] `bucketName` `bucketType`

Required information:

- `bucketName` - Your chosen bucket name.
  - The bucket name must follow the normal bucket guidelines.
- `bucketType` - Choice of bucket type.
  - Public
    - Applications will **not** need an authorization token to download from bucket.
  - Private
    - Application will need an authorization token to download from bucket.

Notes:

In the B2 docs for the CLI, the required `bucketType` argument show `[allPublic | allPrivate]` instead and is wrapped in square brackets, but the choice between `allPublic` and `allPrivate` is required. You must choose one or the other and not both.

Optional arguments:

The following optional arguments need values in JSON format. There are some rules for JSON formatting.

- If adding more than one VARIABLE:VALUE pair, it must be separated with a comma.

- If a VALUE is to be treated as an integer (number), it shouldn't be in double quotes. If a VALUE is to be treated as a string (text), it should be in double quotes

---

[--bucketInfo BUCKETINFO] - These are user chosen and user defined values. They can be named anything and be anything the user wants.

BUCKETINFO Format:

The BUCKETINFO part must be replaced with a JSON formatted as '{"VARIABLE": "VALUE"}'

If a VALUE is to be treated as an integer (number), it shouldn't be in double quotes. If a VALUE is to be treated as a string (text), it should be in double quotes

Example:

b2 create-bucket --bucketInfo '{"people": "cool","pizza": "yes"}' `mattclibucket2` `allPublic`

For more information: https://www.backblaze.com/b2/docs/buckets.html

---

[--corsRules CORSRULES] - This is to set CORS Rules (Cross-Origin Resource Sharing).

CORSRULES Format:

The CORSRULES part must be replaced with a JSON formatted as '[{"VARIABLE": "VALUE", "VARIABLE2": "VALUE2"}]'

Since this JSON requires multiple VARIABLE:VALUE pairs, it must have those square braces [ ] in the wrapping.

Example:

b2 create-bucket --corsRules '[{"corsRuleName":"downloadFromAnyOrigin", "allowedOrigins": ["https"], "allowedHeaders": ["range"], "allowedOperations": ["b2_download_file_by_id", "b2_download_file_by_name"], "exposeHeaders": ["x-bz-content-sha1"], "maxAgeSeconds": 3600}]' `mattybucket` `allPublic`

For more CORS Rules information: https://www.backblaze.com/b2/docs/cors_rules.html

---

[--lifecycleRules LIFECYCLERULES] - This is to set Lifecycle Rules.

LIFECYCLERULES Format:

The LIFECYCLERULES part must be replaced with a JSON formatted as '[{"VARIABLE": "VALUE", "VARIABLE2": "VALUE2"}]'

Since this JSON requires multiple VARIABLE:VALUE pairs, it must have those square braces [ ] in the wrapping.

Example:

b2 create-bucket --lifecycleRules '[{"daysFromHidingToDeleting": 1, "daysFromUploadingToHiding": null, "fileNamePrefix": "chunks/"}]' `mattybux` `allPublic`

For more Lifecycle Rules information: https://www.backblaze.com/b2/docs/lifecycle_rules.html

---

Default Server-Side Encryption:

The option [--defaultServerSideEncryption {SSE-B2,none}] will set Server-Side Encryption for the bucket.

- If the value chosen is SSE-B2 it will enable Server-Side Encryption for the bucket. This will not encrypt non-encrypted objects that are already in the bucket.
- If the value chosen is none it will disable Server-Side Encryption for the bucket. This will not decrypt encrypted objects that are already in the bucket.

The option [--defaultServerSideEncryptionAlgorithm {AES256}] doesn't have an effect on the call as enabling Server-Side Encryption with --defaultServerSideEncryption SSE-B2 will set --defaultServerSideEncryptionAlgorithm to AES256 automatically.

Example:

b2 create-bucket --defaultServerSideEncryption SSE-B2 `machu-bux2` `allPublic`

For more information: Server-Side Encryption

Output:

The call will give back the brand new created bucket ID. Adding the optional arguments will not change the output.

Input	Output
b2 create-bucket `mattsbucko` `allPublic`	

	Link to screenshot

b2 create-key

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

Required information:

- `keyName` - Name of the key.
  - Key names are required, can contain letters, numbers, and "-", and are limited to 100 characters.
- `capabilities` - User chosen capabilities (aka permissions) for the application key.
  - Multiple capabilities can be listed separated by a comma and no space.

Notes:

When making this call, the current application key must possess read AND write capabilities in order to make this call. Although the specific permission is called `writeKeys`, it is a permission specific to being both read and write.

Permissions:

The list of possible permissions as stated on https://www.backblaze.com/b2/docs/b2_create_key.html is not the full list of permissions that you can list in this call. You are able to list any and all possible permissions ranging from ones required for object lock, server-side encryption as well as `listAllBucketNames` which is required to use the S3 compatible API when creating an application key restricted to a single bucket with [--bucket BUCKET].

Optional arguments:

[--bucket BUCKET] - Like the above, the list of possible permissions as stated on https://www.backblaze.com/b2/docs/b2_create_key.html is not the full list of permissions that you can list in this call.

[--namePrefix NAMEPREFIX] restricts file access to files whose names start with the prefix or file path.

[--duration DURATION] will set a timer for the application key that starts after the key is created. When the timer runs out, the key will be deleted and will no longer be usable. If this is left out of the call, the key will not expire and will not be deleted until the user manually deletes it through the CLI or the web UI.

Output:

The call will give back both the keyId and applicationKey. Optional arguments will not change the output of this call.

Input	Output

b2 create-key `newkey readBucketEncryption, listBuckets`	`matto@MacBook-Pro ~ % b2 create-key newkey readBucketEncryption,listBuckets`
	`0002afe34c8bd690000000064 K000Ja63hKkeDyEUbaC0kg3GW2WccVc`
	Link to screenshot

b2 delete-bucket

b2 delete-bucket [-h] `bucketName`

Required information:

- `bucketName` - Name of the bucket that will be deleted.

Notes:

The bucket that will be deleted must be empty and free of any files.

The placeholder file called ".DS_Store" left over from the cli command `b2 sync` with optional [--delete] and [--allowEmptySource] will count as a file that prevents the bucket from being deleted. It can easily be deleted in the web UI or with the `b2 delete-file-version` so that you can delete the bucket.

Input	Output
b2 delete-bucket `mattyooo`	When successful, this call will not give any response.
b2 delete-bucket `himatt`	`matto@MacBook-Pro ~ % b2 delete-bucket himatt` `ERROR: Cannot delete non-empty bucket (cannot_delete_non_empty_bucket)`

This call failed because the bucket had files in it.	 Link to screenshot

b2 delete-file-version

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

Required information:

- `fileId` - File ID for the file that will be deleted.
  - This can be found with `b2 ls` with the optional [--long] argument.

Notes:

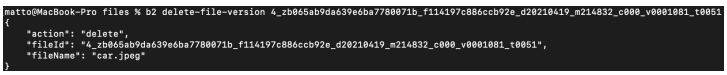
If there is a specific version of a file that you want to delete, each version of the same file will have its own unique fileId.

Optional arguments:

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.

Output:

Optional arguments will not change the output of this call.

Input	Output
b2 delete-file-version `4_zb065ab9da639e6ba7780071b_f114197c886ccb92e_d20210419_m214832_c000_v0001081_t0051`	 Link to screenshot

b2 delete-key

b2 delete-key [-h] `applicationKeyId`

Required information:

- `applicationKeyId` - The keyId from the pair you want to delete.

Output:

The keyId will be returned if the call is successful.

Input	Output
b2 delete-key `0002afe34c8bd690000000066`	 Link to screenshot

b2 download-file-by-id

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

Required information:

- `fileId` - File ID for the file that will be downloaded.
  - This can be found with `b2 ls` with the optional [--long] argument.
- `localFileName` - User chosen local file name.
  - Does not have to match the original.

Notes:

This call will download the file to the location that the CLI is currently in. However, a full file path plus file name can take the place of localFileName.

When entering the `b2FileName`, a prefix can be added as a folder path if the file is nested in one or more subfolders.

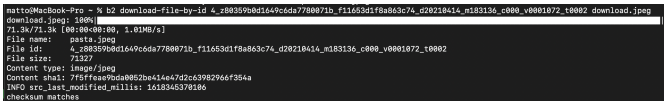
When entering the `localFileName` a prefix can be added as a folder path. If the folder path does not exist yet, a new folder or new set of folders will be created when the call is sent.

Optional arguments:

The argument [--noProgress] will simply do the same thing but will not show a progress bar for the download.

Output:

The output will give back a progress bar and will give back information about the uploaded file if successful.

Input	Output
b2 download-file-by-id `4_z80359b0d1649c6da7780071b_f11653d1f8a863c74_d20210414_m183136_c000_v0001072_t0002 download.jpeg`	 Link to screenshot
b2 download-file-by-id `4_z80359b0d1649c6da7780071b_f11653d1f8a863c74_d20210414_m183136_c000_v0001072_t0002 /Users/matto/desktop/download.jpeg`	 Link to screenshot

b2 download-file-by-name

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

Required information:

- `bucketName` - Name of the bucket that the file will be downloaded from.
- `b2FileName` - The full path plus file name that will be downloaded.
- `localFileName` - User chosen local file name.
  - Does not have to match the original.

Notes:

This call will download the file to the location that the CLI is currently in. However, a full file path can be added as a prefix to the `localFileName`.

When entering the `b2FileName`, a prefix can be added as a folder path if the file is nested in one or more subfolders.

When entering the `localFileName` a prefix can be added as a folder path. If the folder path does not exist yet, a new folder or new set of folders will be created when the call is sent.

Optional arguments:

The argument [--noProgress] will simply do the same thing but will not show a progress bar for the download.

Output:

The output will give back a progress bar and will give back information about the uploaded file if successful.

Input	Output

b2 download-file-by-name `matt-cli-bucket pasta.jpeg pastadl.jpeg`	`matto@MacBook-Pro ~ % b2 download-file-by-name matt-cli-bucket pasta.jpeg pastadl.jpeg`
	`pastadl.jpeg: 100%| | 71.3k/71.3k [00:00<00:00, 571kB/s]`
	`File name: pasta.jpeg`
	`File id: 4_z80359b0d1649c6da7780071b_f1046c93dff9e6a59_d20210414_m205117_c000_v0001075_t0054`
	`File size: 71327`
	`Content type: image/jpeg`
	`Content sha1: 7f5ffeae9bda0052be414e47d2c63982966f354a`
	`INFO src_last_modified_millis: 1618345370106`
	`checksum matches`

	Link to screenshot
b2 download-file-by-name `matt-cli-bucket dog.jpeg dog.jpeg` This call failed because the b2FileName was just the file name. dog.jpeg in the bucket is in example/animals/dog.jpeg	 Link to screenshot

b2 get-account-info

b2 clear-account [-h]

No required information.

Notes:

Despite the name of the call, this will not give back any information about the Backblaze account. This will actually only give back information about the applicationKey, the current session's authentication token, its permissions, apiUrl and downloadUrl.

Input	Output

b2 get-account-info
 	
After successfully calling `b2 authorize-account`
	matto@MacBook-Pro ~ % b2 get-account-info
	{
	"accountAuthToken": "4_0002afe34c8bd690000000068_019cb04c_541d63_acct_DQ15Mk3_v6WKry7Kgm6Z-BOPokQ=",
	"accountId": "2[redacted]9",
	"allowed": {
	"bucketId": null,
	"bucketName": null,
	"capabilities": [
	"listKeys",
	"writeKeys",
	"deleteKeys",
	"listBuckets",
	"writeBuckets",
	"deleteBuckets",
	"readBuckets",
	"listFiles",
	"readFiles",
	"shareFiles",
	"writeFiles",
	"deleteFiles",
	"readBucketEncryption",
	"writeBucketEncryption",
	"bypassGovernance",
	"readBucketRetentions",
	"writeBucketRetentions",
	"readFileRetentions",
	"writeFileRetentions",
	"readFileLegalHolds",
	"writeFileLegalHolds"
	],
	"namePrefix": null
	},
	"apiUrl": "https://api000.backblazeb2.com",
	"applicationKey": "K[redacted]VI",
	"downloadUrl": "https://f000.backblazeb2.com"
	}
	matto@MacBook-Pro ~ %
	
	Link to screenshot

b2 get-account-info Without successfully calling `b2 authorize-account` or after calling `b2 clear-account`.	matto@MacBook-Pro ~ % b2 get-account-info ERROR: Missing account data: 'NoneType' object is not subscriptable Use: b2 authorize-account or provide auth data with "B2_APPLICATION_KEY_ID" and "B2_APPLICATION_KEY" environment variables Link to screenshot

b2 get-bucket

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

Required information:

- `bucketName` - Name of bucket you want to get information for.

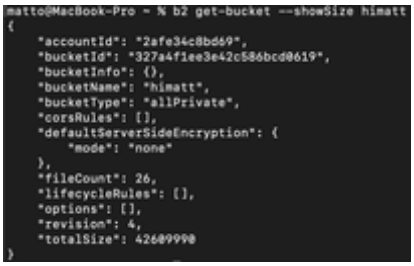
Notes:

This is a lot like `b2 get-file-info` in that it returns information about the bucket.

Optional arguments:

[--showSize] - This will add `totalSize` to the bottom of the list and will display the total size of the bucket in bytes.

Output:

Input	Output
b2 get-bucket --showSize `himatt`	 Link to screenshot

b2 get-file-info

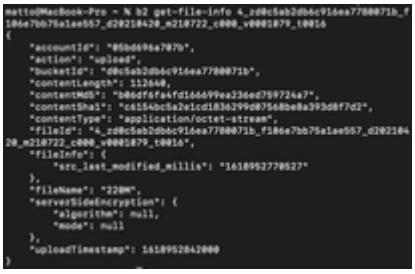
b2 get-file-info [-h] `fileId`

Required information:

- `fileId` - File ID for the file that will be deleted.
  - This can be found with `b2 ls` with the optional [--long] argument.

Notes:

This will give general information about the file stored in B2.

Input	Output
b2 get-file-info `4_zd0c5ab2db6c916ea7780071b_f106e7bb75a1ae557_d20210420_m210722_c000_v0001079_t0016`	 Link to screenshot

b2 get-download-auth

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

Required information:

- `bucketName` - Name of the bucket that the authorization token is for.

Notes:

This will return an authorization token that can be passed to `b2_download_file_by_name` (link; not the CLI version) through the `Authorization` header and allow another user or machine to download a file or files from their bucket.

Optional Arguments:

[--prefix PREFIX] - You can add a file path in the bucket that will restrict the authorization token to only download files from that specific folder path.

- The prefix does not have to match an existing folder or set of folders within the bucket. If done this way, the call will still return an authorization token that is only allowed to download objects within the non-existent folder or set of folders. In other words, this token will not have access to any objects in the bucket if the prefix does not exist.

[--duration DURATION] - Specifies how long the auth token is valid for in seconds.

Output:

All variations of this call will only return the authorization token.

Input	Output
b2 get-download-auth --prefix snapshots `himatt`	`matto@MacBook-Pro ~ % b2 get-download-auth --prefix snapshots himatt 3_20210502213522_719dcaa8a38929bba28d401a_25f7ac9f8515b0db32b107b1233134b21f9dc703_000_20210503213522_0009_dnld` Link to screenshot

b2 get-download-url-with-auth

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

Required information:

- `bucketName` - Name of bucket the url will download the file from.
- `fileName` - Name of the file including file prefix/path if applicable.

Notes:

Useful for allowing a user to download a file from a private bucket.

Similar to [--prefix PREFIX] in `b2 get-download-auth`, the fileName does not need to exist for the CLI to return a url. When the URL is accessed in an attempt to download a file that **does not exist** in the bucket but is specified with fileName, the browser will give back a 404 error.

Optional Argument:

[--duration DURATION] - Specifies how long the authorized URL is valid for in seconds.

Output:

All variations of this call will only return the authorized download URL.

Input	Output
b2 get-download-url-with-auth `himatt hihello2.rtf`	`matto@MacBook-Pro ~ % b2 get-download-url-with-auth himatt hihello2.rtf https://f000.backblazeb2.com/file/himatt/hihello2.rtf?Authorization=3_20210502215215_752fb88bcedf549f0c817a02_84764a56be63f265798c6b53208544c1453be43e_000_20210503215215_0012_dnld` Link to screenshot

b2 hide-file

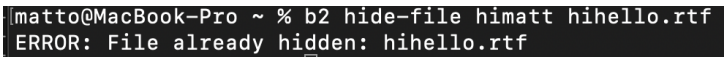
b2 hide-file [-h] `bucketName fileName`

Required information:

- `bucketName` - Name of the bucket the file to be hidden is stored in.
- `fileName` - Name of the file including file prefix/path if applicable.

Notes:

This will download and then upload the file and will be considered a different version of the same file. The original will still exist.

Input	Output
b2 hide-file `himatt hihello.rtf`	 Link to screenshot
b2 hide-file `himatt hihello.rtf` If the file is already hidden, this will show up.	 Link to screenshot

b2 list-buckets

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

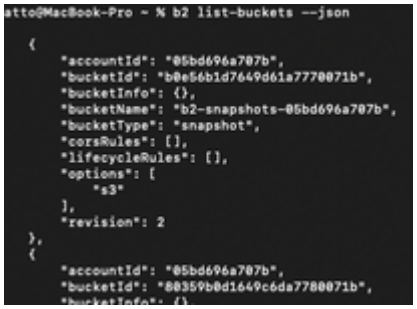
No required information.

Notes:

Unlike B2 docs for this call, optional arguments such as *bucketId*, *bucketName* and *bucketTypes* are not an option in the CLI. These options will filter in whatever buckets match these arguments. In the CLI, all buckets will be given back in the response.

Optional arguments:

The CLI version has an optional argument [--json] which formats the list of buckets in a machine-readable output and will also include more bucket information.

Input	Output
b2 list-buckets	 Link to screenshot
b2 list-buckets --json	 Link to screenshot

b2 list-keys

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

No required information.

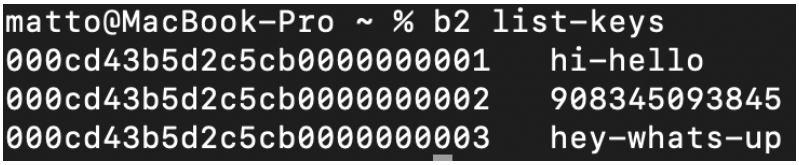
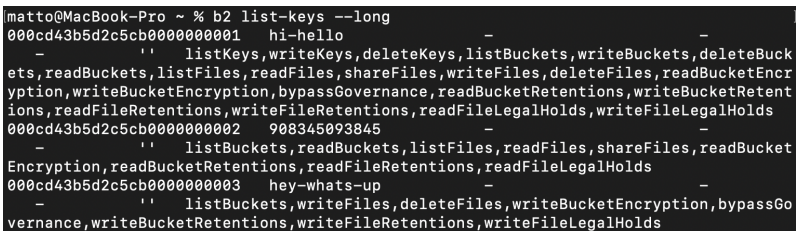
Notes:

This will list all keyIds created in account. This will NOT list applicationKeys

Optional argument:

The argument [--long], on top of listing all keyIds, will also list bucket restriction, prefix restriction, duration, as well as all capabilities/permissions.

Output:

Input	Output
b2 list-keys	 Link to screenshot
b2 list-keys --long	 Link to screenshot

b2 list-parts

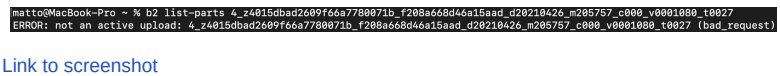
b2 list-parts [-h] `largeFileId`

Required information:

- `largeFileId` - File ID of the large file whose individual parts you want to list.
  - This can be found with `b2 list-unfinished-large-files`.

Notes:

This will only work for a large file upload that has not been finished or canceled.

Input	Output
b2 list-parts `4_z4015dbad2609f66a7780071b_f208a668d46a15aad_d20210426_m205757_c000_v0001080_t0027` This call resulted in an ERROR because the fileId belonged to a file that has already finished uploading.	 Link to screenshot

b2 list-unfinished-large-files

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

Required information:

- `bucketName` - Name of the bucket whose unfinished large files will be listed.

Output:

It will list the fileId, the fileName, contentType, and lastModified for each unfinished large file.

Input	Output

b2 list-unfinished-large-files `mattta`	`matto@MacBook-Pro ~ % b2 list-unfinished-large-files mattta`
	`4_z4015dbad2609f66a7780071b_f201b37dab303575e_d20210426_m191704_c000_v0001158_t0041 220m application/octet-stream src_last_modified_millis=1618952922631`
	Link to screenshot

b2 ls

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

Required information:

- `bucketName` - Name of the bucket whose contents will be listed.
- [`folderName`] - If there are contents nested in "folders", those folder paths will be needed as well.

With `bucketName` alone, this call will list the contents at the top level of the bucket.

Optional arguments:

If there are desired contents nested in a folder, the optional arguments [--recursive] and/or [folderName] will need to be included in the call.

- When making the call with [--recursive] without a specified [folderName], it will list everything nested in each folder at the top level.
- When making the call with [folderName], it must either be a single folder name or a full folder path. It will list just the contents in the top level of the specified folder or ending folder of the full folder path.
- When making the call with both [--recursive] and [folderName], it will list the contents at the top level and everything nested in the specified folder or the ending folder of the full folder path.
- If included, the [folderName] must be *after* bucketName

If you need further information about the contents in the bucket or bucket/folder, include the argument [--long]. This will display file ID, upload date /time, file size and file name.

Bucket contents for the examples below:

- matt-cli-bucket
  - /example/
    - animals/
      - dog.jpeg
  - pasta.jpeg

Output:

Input	Output

b2 ls `matt-cli-bucket`	`matto@MacBook-Pro ~ % b2 ls matt-cli-bucket`
	`example/`
	`pasta.jpeg`
	Link to screenshot

b2 ls `matt-cli-bucket example`	`matto@MacBook-Pro ~ % b2 ls matt-cli-bucket example`
	`example/animals/`
	Link to screenshot

b2 ls `matt-cli-bucket example /animals`	`matto@MacBook-Pro ~ % b2 ls matt-cli-bucket example/animals`
	`example/animals/dog.jpeg`
	Link to screenshot

b2 ls --recursive `matt-cli-bucket`	`matto@MacBook-Pro ~ % b2 ls --recursive matt-cli-bucket`
	`example/animals/dog.jpeg`
	`pasta.jpeg`
	Link to screenshot

b2 ls `matt-cli-bucket` --long
 	
Note, since a [folderName] was not included, it will not list the stuff nested in /example/. It does, however, list the folder example
	`matto@MacBook-Pro ~ % b2 ls matt-cli-bucket --long`
	`-  -  -  -  0  example/`
	`4_z80359b0d1649c6da7780071b_f11653d1f8a863c74_d20210414_m183136_c000_v0001072_t0002  upload  2021-04-14  18:31:36  71327  pasta.jpeg`
	Link to screenshot

itself since it is at the top level of the bucket.	

b2 make-url

b2 make-url [-h] `fileId`

Required information:

- `fileId` - File ID for the file that the URL will link to.

Notes:

This will create a download link to the file with the `fileId` in the URL. The URL will be the same as the “Native URL” in the file info when browsing the bucket through the web UI. Here is an example.

If the bucket is private, this will successfully create a download link to the file but the link cannot be accessed without authorization. Here is an example of what you would see if you try to access a file from a private bucket.

Output:

Input	Output

b2 make-url `4_z327a4f1ee3e42c586bcd0619_f11811702e1d33ee9_d20191231_m172906_c000_v0001064_t0027`	`matto@MacBook-Pro ~ % b2 make-url 4_z327a4f1ee3e42c586bcd0619_f11811702e1d33ee9_d20191231_m172906_c000_v0001064_t0027`
	`https://f000.backblazeb2.com/b2api/v2/b2_download_file_by_id?fileId=4_z327a4f1ee3e42c586bcd0619_f11811702e1d33ee9_d20191231_m172906_c000_v0001064_t0027`
	
	Link to screenshot

b2 make-friendly-url

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

Required information:

- `bucketName` - Name of bucket that is storing the file to share.
- `fileName` - Name of the file including file prefix/path if applicable.

Notes:

This will create a download link to the file with the file `bucketName` and `fileName` in the URL. The URL will be the same as the “Friendly URL” in the file info when browsing the bucket through the web UI. Here is an example.

If the bucket is private, this will successfully create a download link to the file but the link cannot be accessed without authorization. Here is an example of what you would see if you try to access a file from a private bucket.

Output:

Input	Output

b2 make-friendly-url `himatt 10.mp4`	`matto@MacBook-Pro ~ % b2 make-friendly-url himatt 10.mp4`
	`https://f000.backblazeb2.com/file/himatt/10.mp4`
	
	Link to screenshot

b2 sync

b2 sync [-h] [--noProgress] [--dryRun] [--allowEmptySource] [--excludeAllSymlinks] [--threads THREADS] [--compareVersions {none, modTime,size}] [--compareThreshold MILLIS] [--excludeRegex REGEX] [--includeRegex REGEX] [--excludeDirRegex REGEX] [--excludeIfModifiedAfter TIMESTAMP] [--destinationServerSideEncryption {SSE-B2}] [--destinationServerSideEncryptionAlgorithm {AES256}] [--skipNewer | --replaceNewer] [--delete | --keepDays DAYS] `source destination`

Required information:

- `source` - The source folder that contents will be copied from.
- `destination` - The destination folder to which the contents will be copied.

Notes:

This call can be used to sync in both directions:

- From bucket/folder path to local folder.
  - This will be a download of the bucket/folder path contents to a local folder.
- From local folder to bucket/folder path.
  - This will be an upload of the local folder contents to a bucket/folder path.

One of the locations have to be a local folder and the other has to be a b2 location. For the B2 location, since this call can work either to or from a bucket, the path must start with `b2://` to denote a network path to the bucket.

When entering the `destination`, a prefix can be added as a folder path. If the folder path does not exist yet, a new folder or new set of folders will be created when the call is sent.

Optional arguments:

This call has a LOT of optional arguments, so these will be split into several sections.

[--noProgress] will simply do the same thing but will not show a progress for the upload or download.

[--dryRun] will simulate the syncing process without actually uploading or downloading data.

[--excludeAllSymlinks] will skip symlinks when syncing from a local source.

Deletions at the destination:

[--delete | --keepDays DAYS]

- --delete will delete whatever data already exists in the destination folder. If this argument is not included, the contents of the source will be coped and **added** to the destination.
- --keepDays DAYS will delete **versions** of any file older than the specified age in DAYS.

[--allowEmptySource] will respect an empty folder and the contents from the destination will be deleted if the source is indeed empty.

> These first two are a helpful way to clear a bucket entirely. In which case, you can sync an empty folder to a bucket with the options --allowEmptySource and --delete. More information on using `b2 sync` to delete the contents of a bucket can be found here: https://help.backblaze.com/hc/en-us/articles/225556127-How-Can-I-Easily-Delete-All-Files-in-a-Bucket-
>
> When you use these two optional arguments to delete stuff in a folder or the bucket itself, it will also upload a placeholder file called ".DS_Store".

[--skipNewer | --replaceNewer] - 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.

- --skipNewer will tell the sync job to ignore files with a newer modification time at the `destination`.
- --replaceNewer will tell the sync job to replace files with a newer modification time at the `destination` with the files with an older modification time at the `source`.

> To make the `destination` EXACTLY match the `source`, you can use the option --replaceNewer and --delete.
>
> b2 sync --delete --replaceNewer `source destination`

> To make the destination match the source, but retain previous versions for 30 days:
>
> b2 sync --keepDays 30 --replaceNewer `source destination`

Sync Exclusions

[--excludeRegex REGEX] can be used to specify a "regular expression" to exclude certain files, file types or folders from being uploaded or downloaded. The regular expression should be formatted a certain way.

[--includeRegex REGEX] can be used to have --excludeRegex make exceptions for the rules specified in its REGEX.

[--excludeDirRegex REGEX] - Folders excluded by --excludeDirRegex will not be included even if it matches a REGEX specified by --includeRegex.

REGEX Formatting

macOS	**Files**
	`'(.*/file.txt)'`
	`'(.*/file1.txt)|(.*/file2.txt)'`
	To target file by extension:
	`'(.*/.txt)'`
	To also target files/files by extension at the top level of the folder
	`'(.*/*file.txt)'`
	`'(.*/*.txt)'`
		Folders
		`'(.*/folder)'`
		`'(.*/folder1)|(.*/folder2)'`
		To also target items at the top level of the folder
		`'(.*/*folder)'`

Windows	**Files**
	`'(.*\file.txt)'`
	`'(.*\file1.txt)|(.*\file2.txt)'`
	To target file by extension:
	`'(.*\.txt)'`
	To also target files/files by extension at the top level of the folder
	`'(.*\*file.txt)'`
	`'(.*\*.txt)'`
		Folders
		`'(.*\folder)'`
		`'(.*\folder1)|(.*\folder2)'`
		To also target items at the top level of the folder
		`'(.*\*folder)'`

Notes	• Folders and files can be specified in the same REGEX input.
	• Folder and file names can be partial and everything with that partial name will be targeted.
	• By default, REGEX will not target folders or files in the top level of the source location you are syncing.
	• In order to have REGEX work for folders or files in the top level of the source location, there are a couple ways to achieve this. Some examples:
	  • **Targeting the top level items per REGEX item**
	  Adding the wildcard right after the forward slash will make the sync not target those items at the top level of the source as well as other places the REGEX shows up in the source.
	  `'(.*/*cars)|(.*/*.DS_Store)|(.*/food)'` will target ALL folders called *cars* and ALL files called *.DS_Store*. That goes for everything at the top level as well. Since `(.*/food)` does not have the * right after the forward slash, it will not be excluded if it is at the top level of the source.
	  `'(.*/*cat.jpg)'` will target ALL cat.jpg files (both top and sub level).
	  `'(.*/cat.jpg)'` will target ONLY sub level cat.jpg files.
	  • **Targeting *ONLY* the top level items for the entire REGEX input**
	  Adding square brackets to the REGEX wrapping will make the sync target every item in the REGEX at the top level of the source and will not target the matching items in subfolders. This method does not require any of the REGEX items to have the * wildcard.
	  `'[(.*/cars)|(.*/.DS_Store)|(.*/food)]'` will target only TOP LEVEL folders called *cars*, TOP LEVEL files called *.DS_Store* and TOP LEVEL folders called *food*.

[--excludeIfModifiedAfter TIMESTAMP] - 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 decimal point (e.g. "1367900664.152")

File Comparisons

By default, a file is the same if the name and modification time are the same as a file that already exists in the destination. If a file is the same, it will not be synced.

[--compareVersions {none,modTime,size}] will tell the sync job to determine whether a file is the same based on your choice for this optional argument.

- none: Comparison using the file name only

- modTime: Comparison using the modification time (default)
- size: Comparison using the file size

[--compareThreshold MILLIS] is to be used when --compareVersions is set to either modTime or size. This will tell --compareVersions to do a fuzzy comparison of files within a threshold of all files that already exist in a bucket based on what --compareVersions is set to. If a file's modification time or size is within what is set with --compareThreshold, it will be considered the same and will not be synced.

Example of compareVersions/compareThreshold

We have a bucket with a file dog.jpeg and it is 10 KB or 10000 Bytes plus a few other files that don't change. If the original file is changed and is now 10.05 KB or 10050 Bytes, a sync job with `b2 sync` will determine that this file is different because of its difference in size.

If this 50 byte difference is not big enough for the user to want to upload via `b2 sync`, it can be ignored with --compareVersions if it is set to size and with --compareThreshold if the threshold is set to 100 (bytes). Example of the call is below.

Server-Side Encryption

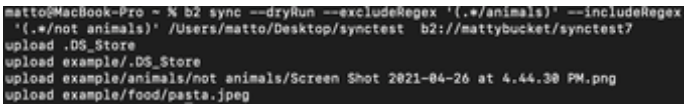
[--destinationServerSideEncryption {SSE-B2}] - When the destination is a B2 bucket, this will encrypt all new files that get uploaded to it. This will not encrypt anything that already exists in the bucket.

> If the intention is to encrypt everything in the bucket with this call, it is better to creating a new bucket and then syncing the local folder to the new bucket. Or even a new folder within the bucket and resyncing the local source to the new folder. Everything newly uploaded will be encrypted.

[--destinationServerSideEncryptionAlgorithm {AES256}] - This option is not necessary as calling the sync job with --destinationServerSideEncryption

Note: The output of the sync job when using --destinationServerSideEncryption will look exactly the same as a standard sync job.

Output:

Input	Output
b2 sync `/Users/matto/Desktop/files b2://mattttata/synctest`	 Link to screenshot
b2 sync `b2://matt-cli-bucket /Users/matto/Desktop/synctest`	 Link to screenshot
b2 sync --dryRun --excludeRegex '(.*/Screen Shot)|(.*/.DS_Store)' `/Users /matto/Desktop/synctest b2://mattybucket/synctest7` This excludes objects with a specific file path. Can be full or partial paths. Also works for files.	 Link to screenshot
b2 sync --dryRun --excludeRegex '(.*/animals)' --includeRegex '(.*/not animals)' `/Users/matto/Desktop/synctest b2://mattybucket /synctest7` This does the same as the above but --includeRegex is used to make an exception for a specific folder. Also works for files.	 Link to screenshot
The destination in this example has a couple files that are unchanged, and a file called *bread.jpeg* that is 40,100 bytes. The source currently has that *bread.jpeg* file but it was modified and is now 39,300 bytes. If we run: • b2 sync --dryRun `/Users/matto/Desktop/files\ 2 b2://himatt/test0` It will want to upload the modified *bread.jpeg* since it is different in size. --- However, if you use --compareVersions and --compareThreshold this difference can be ignored and the modified *bread.jpeg* file will not be	 Link to screenshot  Link to screenshot

uploaded. The difference between the files is 800 bytes, so we can set the threshold to 1000 bytes so that the newer and smaller version is not accounted for in the sync job.

The call looks like this:

- b2 sync --dryRun --compareVersions size --compareThreshold 1000 `/Users/matto/Desktop/files\ 2 b2://himatt/test0`

Notice that the sync job no longer wants to pick up the modified *bread.jpeg*.

b2 update-bucket

b2 update-bucket [-h] [--bucketInfo BUCKETINFO] [--corsRules CORSRULES] [--lifecycleRules LIFECYCLERULES] [--defaultServerSideEncryption {SSE-B2,none}] [--defaultServerSideEncryptionAlgorithm {AES256}] `bucketName bucketType`

Required information:

- `bucketName` - Name of the bucket that will be updated.
- `bucketType` - Choice of bucket type.
  - Public
    - `allPublic` - Applications will not need an authorization token to download from bucket.
  - Private
    - `allPrivate` - Application will need an authorization token to download from bucket.

Notes:

In the B2 docs for the CLI, the required `bucketType` argument show `[allPublic | allPrivate]` instead and is wrapped in square brackets, but the choice between `allPublic` and `allPrivate` is required. You must choose one or the other and not both.

Optional arguments:

The following optional arguments need values in JSON format. There are some rules for JSON formatting.

- If adding more than one VARIABLE:VALUE pair, it must be separated with a comma.
- If a VALUE is to be treated as an integer (number), it shouldn't be in double quotes. If a VALUE is to be treated as a string (text), it should be in double quotes

---

[--bucketInfo BUCKETINFO] - These are user chosen and user defined values. They can be named anything and be anything the user wants.

BUCKETINFO Format:

The BUCKETINFO part must be replaced with a JSON formatted as '{"VARIABLE": "VALUE"}'

If a VALUE is to be treated as an integer (number), it shouldn't be in double quotes. If a VALUE is to be treated as a string (text), it should be in double quotes

For more information: https://www.backblaze.com/b2/docs/buckets.html

---

[--corsRules CORSRULES] - This is to set CORS Rules (Cross-Origin Resource Sharing).

CORSRULES Format:

The CORSRULES part must be replaced with a JSON formatted as '[{"VARIABLE": "VALUE", "VARIABLE2": "VALUE2"}]'

Since this JSON requires multiple VARIABLE:VALUE pairs, it must have those square braces [ ] in the wrapping.

For allowedHeaders, allowedOrigins and allowedOperations, the variables must be wrapped in [" "]. Like this: "allowedHeaders": ["range"]

For more CORS Rules information: https://www.backblaze.com/b2/docs/cors_rules.html

---

[--lifecycleRules LIFECYCLERULES] - This is to set Lifecycle Rules. For more information: https://www.backblaze.com/b2/docs/lifecycle_rules.html

LIFECYCLERULES Format:

The LIFECYCLERULES part must be replaced with a JSON formatted as '[{"VARIABLE": "VALUE", "VARIABLE2": "VALUE2"}]'

Since this JSON requires multiple VARIABLE:VALUE pairs, it must have those square braces [ ] in the wrapping.

For more Lifecycle Rules information: https://www.backblaze.com/b2/docs/lifecycle_rules.html

Default Server-Side Encryption:

The option [--defaultServerSideEncryption {SSE-B2,none}] will set Server-Side Encryption for the bucket.

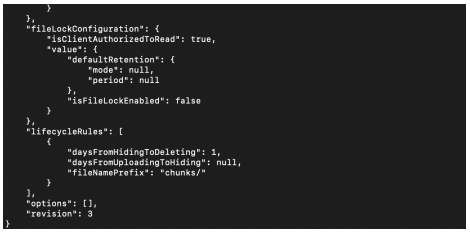
- If the value chosen is SSE-B2 it will enable Server-Side Encryption for the bucket. This will not encrypt non-encrypted objects that are already in the bucket.
- If the value chosen is none it will disable Server-Side Encryption for the bucket. This will not decrypt encrypted objects that are already in the bucket.

The option [--defaultServerSideEncryptionAlgorithm {AES256}] doesn't have an effect on the call as enabling Server-Side Encryption with --defaultServerSideEncryption SSE-B2 will set --defaultServerSideEncryptionAlgorithm to AES256 automatically.

For more information: Server-Side Encryption

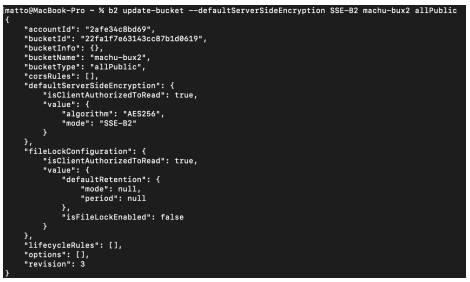
Output:

Input	Output
b2 update-bucket `himatt allPublic`	 Link to screenshot
b2 update-bucket --bucketInfo '{"people": "cool","pizza": "yes"}' `mattclibucket2 allPublic`	 Link to screenshot
b2 update-bucket --corsRules '[{"corsRuleName":"downloadFromAnyOrigin", "allowedOrigins": ["https"], "allowedHeaders": ["range"], "allowedOperations": ["b2_download_file_by_id", "b2_download_file_by_name"], "exposeHeaders": ["x-bz-content-sha1"], "maxAgeSeconds": 3600}]' `mattybucket allPublic`	 Link to screenshot
b2 update-bucket --lifecycleRules '[{"daysFromHidingToDeleting": 1, "daysFromUploadingToHiding": null, "fileNamePrefix": "chunks/"}]' `mattybux allPublic`	



Link to screenshot

b2 update-bucket --defaultServerSideEncryption SSE-B2 `machu-bux2` `allPublic`



Link to screenshot

b2 upload-file

b2 upload-file [-h] [--noProgress] [--quiet] [--contentType CONTENTTYPE] [--minPartSize MINPARTSIZE] [--sha1 SHA1] [--threads THREADS] [--info INFO] [--destinationServerSideEncryption {SSE-B2}] [--destinationServerSideEncryptionAlgorithm {AES256}] `bucketName` `localFilePath` `b2FileName`

Required information:

- `bucketName` - Name of the bucket that a file will be uploaded to.
- `localFilePath` - Full file path + file name of the file that will be uploaded.
- `b2FileName` - User chosen file name.
  - Does not have to match the original.

Notes:

Unlike the normal B2 docs call, the CLI version of this call will not require the *authorization token* nor the *upload URL* input. It will handle the `b2 get-upload-url` call and plug in the URL and upload auth token automatically.

When entering the `b2FileName`, a prefix can be added as a folder path. If the folder path does not exist yet, a new folder or new set of folders will be created when the call is sent.

Optional arguments:

[--noProgress] will simply do the same thing but will not show a progress bar for the download.

[--quiet] will do the same thing but will not list either of the URLs for the object in the bucket.

[--contentType CONTENTTYPE] will set the content-type for the file uploaded. If this is left out of the upload call, it will default according to the file's extension.

[--sha1 SHA1] allows you to tell the call the sha1 of the file to be uploaded. Without this argument, the call will automatically calculate it for you. This is useful to make the processing and computing for many upload calls take less time in total. If making this call one at a time, it's easier to let the call do it for you.

[--info INFO] - These are variables and values chosen and defined by the user.

- This is to be formatted as `VARIABLE=VALUE`. Example below.
- If more than one `VARIABLE=VALUE` is to be specified, and additional --info entry must be made. Example below.

Large Files:

Large files can be from 5MB to 10TB.

If the file being uploaded is 200MB or larger, the CLI will default to treating the file as a large file and upload it in multiple threads. If not specified by the optional [--threads THREADS], the CLI will default to 10 threads. Using the optional [--threads THREADS] argument on a file smaller than

200MB will not have an effect and the CLI will ignore that argument. You can set the size of the parts with [--minPartSize MINPARTSIZE]. If left out, the CLI will choose the “recommendedPartSize” for you.

A file (larger than 200MB) will be broken up into parts and each part will be uploaded on its own thread to reduce the amount of time taken to upload the file in whole.

When uploading a large file in parts, the CLI will not look any different as it does when uploading a small file under 200MB.

If you want to test uploading a large file but don't have one large enough.

You can create a dummy file of any size by running this command:

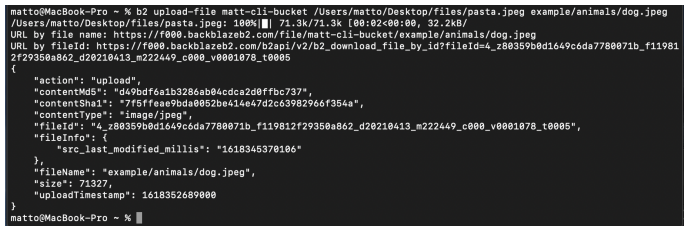
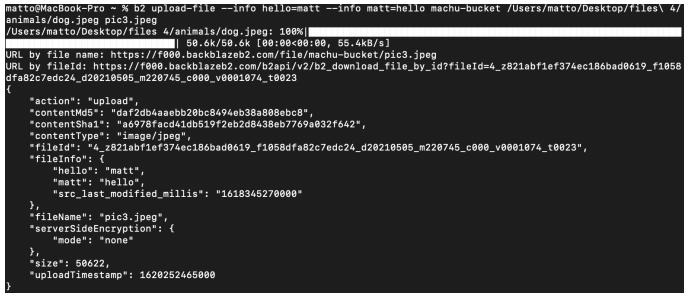
```
mkfile -n <size> <file name>
```

Example:

```
mkfile -n 100M TestFile100M
```

Output:

The call will give back information similar to `b2 get-file-info`.

Input	Output
b2 upload-file `matt-cli-bucket /Users/matto/Desktop/files /pasta.jpeg example/animals/dog.jpeg`	 Link to screenshot
b2 upload-file --info hello=matt --info matt=hello `machu-bucket /Users /matto/Desktop/files\ 4/animals/dog.jpeg pic3.jpeg`	 Link to screenshot

b2 update-file-legal-hold

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

Required information:

- fileId - File ID of the file we will toggle legal hold for.
- {on,off} - User’s choice for if legal hold should be on or off.

Notes:

This call will toggle Object Lock on a specified file and it can’t be deleted until it is disabled.

In order to use this command, the bucket itself needs to have `fileLockEnabled=true`. Required capabilities for the application key are: `writeFileLegalHolds` and (if file name is not provided) `readFiles`.

Optional arguments:

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 capability bypassGovernance.

Output:

If the API accepts the call and is successful, it will return nothing.

If the API doesn't accept the call or fails, it will return the error and prompt that pertains to the issue (example below)

Input	Output
b2 update-file-legal-hold `4_z521acf4e63241c987b5d0619_f111ad32d4d6d1cd7_d20210624_m202933_c000_v0001076_t0026` on Since a successful call doesn't pass anything back, the example to the right includes **b2 get-file-info** after.	 Link to screenshot
b2 update-file-legal-hold `4_z223abf4eb334ec486bbd0619_f1022ec06996c501e_d20220710_m202519_c000_v0001059_t0031_u01657484719094` on This failed because the bucket does not have file lock enabled. (no object lock)	 Link to screenshot

b2 update-file-retention

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

Required information:

- fileId - File ID for the file we want to update file retention for.
- {governance,compliance,none} - Choose between `governance`, `compliance` or `none`.

Notes:

This call will set a certain amount of time of Object Lock for a specified file and it can't be deleted for that amount of time.

In order to use this command, the bucket itself needs to have `fileLockEnabled=true`. Required capabilities for the application key are: `writeFileLegalHolds` and (if file name is not provided) `readFiles`.

File retention is synonymous to "Object Lock" "Immutability" and "File Lock"

Optional arguments:

[--retainUntil TIMESTAMP] will lock the file from being deleted from a bucket until the specified time. The timestamp must be an integer representing milliseconds since "epoch".

In addition to the above, if the type of file retention is set to `governance`, in order to disable or shorten file retention, the user must be using an application key with the capability bypassGovernance and the call must also pass the [--bypassGovernance] argument in the call.

If the type of file retention is set to `compliance`, the user will not be able to remove or shorten file retention for the file. The file must reach the end of the retention period set before it can be deleted.

Epoch dates & timestamps:

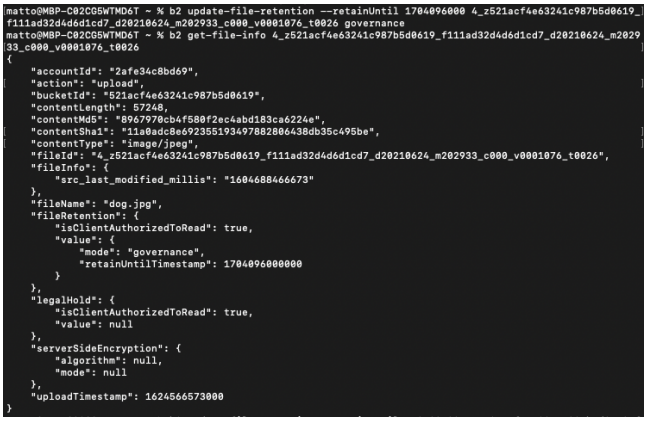
A date in epoch format is an integer that represents milliseconds from January 1, 1970 to a specified date.

For example, if you want a file to be locked until January 1, 2024, you would set --retainUntil 1704096000 where 1704096000 is the number of milliseconds from January 1, 1970 to January 1, 2024. This is a useful tool to calculate a date converted to epoch.

Output:

If the API accepts the call and is successful, it will return nothing.

If the API doesn’t accept the call or fails, it will return the error and prompt that pertains to the issue

Input	Output
b2 update-file-retention --retainUntil 1704096000 `4_z521acf4e63241c987b5d0619_f111ad32d4d6d1cd7_d20210624_m202933_c000_v0001076_t0026 governance` Since a successful call doesn’t pass anything back, the example to the right includes **b2 get-file-info** after.	 Link to screenshot
b2 update-file-retention --retainUntil 170409600 `4_z521acf4e63241c987b5d0619_f111ad32d4d6d1cd7_d20210624_m202933_c000_v0001076_t0026 governance` This call fails because the requested retention time is not in the future.	 Link to screenshot