Dropbox for Python

Release 7.1.0

October 27, 2016

Contents

1 Tutorial **1**

2 SDK Documentation **3**

2.1 `dropbox.oauth` – OAuth . . . 3
2.2 `dropbox.dropbox` – Dropbox . . . 5
2.3 `dropbox.async` – Async . . . 46
2.4 `dropbox.files` – Files . . . 48
2.5 `dropbox.sharing` – Sharing . . . 97
2.6 `dropbox.team` – Team . . . 182
2.7 `dropbox.users` – Users . . . 245
2.8 `dropbox.exceptions` – Exceptions . . . 252

3 Indices and tables **255**

Python Module Index **257**

CHAPTER 1

Tutorial

For a tutorial on how to get started, please visit our developers page.

CHAPTER 2

SDK Documentation

2.1 dropbox.oauth – OAuth

exception dropbox.oauth.**BadRequestException**

Thrown if the redirect URL was missing parameters or if the given parameters were not valid.

The recommended action is to show an HTTP 400 error page.

exception dropbox.oauth.**BadStateException**

Thrown if all the parameters are correct, but there's no CSRF token in the session. This probably means that the session expired.

The recommended action is to redirect the user's browser to try the approval process again.

exception dropbox.oauth.**CsrfException**

Thrown if the given 'state' parameter doesn't contain the CSRF token from the user's session. This is blocked to prevent CSRF attacks.

The recommended action is to respond with an HTTP 403 error page.

class dropbox.oauth.**DropboxOAuth2Flow**(*consumer_key*, *consumer_secret*, *redirect_uri*, *session*, *csrf_token_session_key*, *locale=None*)

OAuth 2 authorization helper. Use this for web apps.

OAuth 2 has a two-step authorization process. The first step is having the user authorize your app. The second involves getting an OAuth 2 access token from Dropbox.

Example:

```
from dropbox import DropboxOAuth2Flow

def get_dropbox_auth_flow(web_app_session):
    redirect_uri = "https://my-web-server.org/dropbox-auth-finish"
    return DropboxOAuth2Flow(
        APP_KEY, APP_SECRET, redirect_uri, web_app_session,
        "dropbox-auth-csrf-token")

# URL handler for /dropbox-auth-start
def dropbox_auth_start(web_app_session, request):
    authorize_url = get_dropbox_auth_flow(web_app_session).start()
    redirect_to(authorize_url)

# URL handler for /dropbox-auth-finish
def dropbox_auth_finish(web_app_session, request):
    try:
```

```
            access_token, user_id, url_state = \
                    get_dropbox_auth_flow(web_app_session).finish(
                        request.query_params)
        except BadRequestException, e:
            http_status(400)
        except BadStateException, e:
            # Start the auth flow again.
            redirect_to("/dropbox-auth-start")
        except CsrfException, e:
            http_status(403)
        except NotApprovedException, e:
            flash('Not approved?  Why not?')
            return redirect_to("/home")
        except ProviderException, e:
            logger.log("Auth error: %s" % (e,))
            http_status(403)
```

finish(*query_params*)

Call this after the user has visited the authorize URL (see `start()`), approved your app and was redirected to your redirect URI.

Parameters **query_params** (*dict*) – The query parameters on the GET request to your redirect URI.

Return type *OAuth2FlowResult*

Raises `BadRequestException` If the redirect URL was missing parameters or if the given parameters were not valid.

Raises `BadStateException` If there's no CSRF token in the session.

Raises `CsrfException` If the `state` query parameter doesn't contain the CSRF token from the user's session.

Raises `NotApprovedException` If the user chose not to approve your app.

Raises `ProviderException` If Dropbox redirected to your redirect URI with some unexpected error identifier and error message.

start(*url_state=None*)

Starts the OAuth 2 authorization process.

This function builds an "authorization URL". You should redirect your user's browser to this URL, which will give them an opportunity to grant your app access to their Dropbox account. When the user completes this process, they will be automatically redirected to the `redirect_uri` you passed in to the constructor.

This function will also save a CSRF token to `session[csrf_token_session_key]` (as provided to the constructor). This CSRF token will be checked on `finish()` to prevent request forgery.

Parameters **url_state** (*str*) – Any data that you would like to keep in the URL through the authorization process. This exact value will be returned to you by `finish()`.

Returns The URL for a page on Dropbox's website. This page will let the user "approve" your app, which gives your app permission to access the user's Dropbox account. Tell the user to visit this URL and approve your app.

class dropbox.oauth.**DropboxOAuth2FlowNoRedirect**(*consumer_key*, *consumer_secret*, *locale=None*)

OAuth 2 authorization helper for apps that can't provide a redirect URI (such as the command-line example apps).

Example:

```
from dropbox import DropboxOAuth2FlowNoRedirect

auth_flow = DropboxOAuth2FlowNoRedirect(APP_KEY, APP_SECRET)

authorize_url = auth_flow.start()
print "1. Go to: " + authorize_url
print "2. Click \"Allow\" (you might have to log in first)."
print "3. Copy the authorization code."
auth_code = raw_input("Enter the authorization code here: ").strip()

try:
    access_token, user_id = auth_flow.finish(auth_code)
except Exception, e:
    print('Error: %s' % (e,))
    return

dbx = Dropbox(access_token)
```

finish(*code*)

If the user approves your app, they will be presented with an "authorization code". Have the user copy/paste that authorization code into your app and then call this method to get an access token.

Parameters **code** (*str*) – The authorization code shown to the user when they approved your app.

Return type *OAuth2FlowNoRedirectResult*

Raises The same exceptions as `DropboxOAuth2Flow.finish()`.

start()

Starts the OAuth 2 authorization process.

Returns The URL for a page on Dropbox's website. This page will let the user "approve" your app, which gives your app permission to access the user's Dropbox account. Tell the user to visit this URL and approve your app.

exception dropbox.oauth.**NotApprovedException**

The user chose not to approve your app.

class dropbox.oauth.**OAuth2FlowNoRedirectResult**(*access_token*, *account_id*, *user_id*)

Authorization information for an OAuth2Flow performed with no redirect.

class dropbox.oauth.**OAuth2FlowResult**(*access_token*, *account_id*, *user_id*, *url_state*)

Authorization information for an OAuth2Flow with redirect.

classmethod **from_no_redirect_result**(*result*, *url_state*)

exception dropbox.oauth.**ProviderException**

Dropbox redirected to your redirect URI with some unexpected error identifier and error message.

The recommended action is to log the error, tell the user something went wrong, and let them try again.

2.2 dropbox.dropbox – Dropbox

class dropbox.dropbox.**Dropbox**(*oauth2_access_token*, *max_retries_on_error=4*, *max_retries_on_rate_limit=None*, *user_agent=None*, *session=None*, *headers=None*, *timeout=30*)

Bases: dropbox.dropbox._DropboxTransport, dropbox.base.DropboxBase

Use this class to make requests to the Dropbox API using a user's access token. Methods of this class are meant to act on the corresponding user's Dropbox.

__init__(*oauth2_access_token*, *max_retries_on_error=4*, *max_retries_on_rate_limit=None*, *user_agent=None*, *session=None*, *headers=None*, *timeout=30*)

Parameters

- **oauth2_access_token** (*str*) – OAuth2 access token for making client requests.
- **max_retries_on_error** (*int*) – On 5xx errors, the number of times to retry.
- **max_retries_on_rate_limit** (*Optional[int]*) – On 429 errors, the number of times to retry. If *None*, always retries.
- **user_agent** (*str*) – The user agent to use when making requests. This helps us identify requests coming from your application. We recommend you use the format "AppName/Version". If set, we append "/OfficialDropboxPythonSDKv2/__version__" to the user_agent,
- **session** (requests.sessions.Session) – If not provided, a new session (connection pool) is created. To share a session across multiple clients, use *create_session()*.
- **headers** (*dict*) – Additional headers to add to requests.
- **timeout** (*Optional[float]*) – Maximum duration in seconds that client will wait for any single packet from the server. After the timeout the client will give up on connection. If *None*, client will wait forever. Defaults to 30 seconds.

auth_token_revoke()

Disables the access token used to authenticate the call.

Return type None

files_alpha_get_metadata(*path*, *include_media_info=False*, *include_deleted=False*, *include_has_explicit_shared_members=False*, *include_property_templates=None*)

Returns the metadata for a file or folder. This is an alpha endpoint compatible with the properties API. Note: Metadata for the root folder is unsupported.

Parameters **include_property_templates** (*Nullable*) – If set to a valid list of template IDs, FileMetadata.property_groups is set for files with custom properties.

Return type *dropbox.files.Metadata*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.AlphaGetMetadataError*

files_alpha_upload(*f*, *path*, *mode=WriteMode('add', None)*, *autorename=False*, *client_modified=None*, *mute=False*, *property_groups=None*)

Create a new file with the contents provided in the request. Note that this endpoint is part of the properties API alpha and is slightly different from *files_upload()*. Do not use this to upload a file larger than 150 MB. Instead, create an upload session with *files_upload_session_start()*.

Parameters

- **f** (*bytes*) – Contents to upload.
- **property_groups** (*Nullable*) – List of custom properties to add to file.

Return type *dropbox.files.FileMetadata*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.UploadErrorWithProperties*

files_copy(*from_path*, *to_path*, *allow_shared_folder=False*, *autorename=False*)

Copy a file or folder to a different location in the user's Dropbox. If the source path is a folder all its contents will be copied.

Parameters

- **allow_shared_folder** (*bool*) – If true, *files_copy()* will copy contents in shared folder, otherwise `RelocationError.cant_copy_shared_folder` will be returned if `from_path` contains shared folder. This field is always true for *files_move()*.
- **autorename** (*bool*) – If there's a conflict, have the Dropbox server try to autorename the file to avoid the conflict.

Return type *dropbox.files.Metadata*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.RelocationError*

files_copy_batch(*entries*, *allow_shared_folder=False*, *autorename=False*)

Copy multiple files or folders to different locations at once in the user's Dropbox. If `RelocationBatchArg.allow_shared_folder` is false, this route is atomic. If on entry failes, the whole transaction will abort. If `RelocationBatchArg.allow_shared_folder` is true, not atomicity is guaranteed, but you will be able to copy the contents of shared folders to new locations. This route will return job ID immediately and do the async copy job in background. Please use *files_copy_batch_check()* to check the job status.

Parameters

- **entries** (*list*) – List of entries to be moved or copied. Each entry is *dropbox.files.RelocationPath*.
- **allow_shared_folder** (*bool*) – If true, *files_copy_batch()* will copy contents in shared folder, otherwise `RelocationError.cant_copy_shared_folder` will be returned if `RelocationPath.from_path` contains shared folder. This field is always true for *files_move_batch()*.
- **autorename** (*bool*) – If there's a conflict with any file, have the Dropbox server try to autorename that file to avoid the conflict.

Return type `dropbox.files.LaunchEmptyResult`

files_copy_batch_check(*async_job_id*)

Returns the status of an asynchronous job for *files_copy_batch()*. If success, it returns list of results for each entry.

Parameters **async_job_id** (*str*) – Id of the asynchronous job. This is the value of a response returned from the method that launched the job.

Return type *dropbox.files.RelocationBatchJobStatus*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: `dropbox.files.PollError`

files_copy_reference_get(*path*)

Get a copy reference to a file or folder. This reference string can be used to save that file or folder to another user's Dropbox by passing it to `files_copy_reference_save()`.

Parameters **path** (*str*) – The path to the file or folder you want to get a copy reference to.

Return type `dropbox.files.GetCopyReferenceResult`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.files.GetCopyReferenceError`

files_copy_reference_save(*copy_reference*, *path*)

Save a copy reference returned by `files_copy_reference_get()` to the user's Dropbox.

Parameters

- **copy_reference** (*str*) – A copy reference returned by `files_copy_reference_get()`.
- **path** (*str*) – Path in the user's Dropbox that is the destination.

Return type `dropbox.files.SaveCopyReferenceResult`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.files.SaveCopyReferenceError`

files_create_folder(*path*, *autorename=False*)

Create a folder at a given path.

Parameters

- **path** (*str*) – Path in the user's Dropbox to create.
- **autorename** (*bool*) – If there's a conflict, have the Dropbox server try to autorename the folder to avoid the conflict.

Return type `dropbox.files.FolderMetadata`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.files.CreateFolderError`

files_delete(*path*)

Delete the file or folder at a given path. If the path is a folder, all its contents will be deleted too. A successful response indicates that the file or folder was deleted. The returned metadata will be the corresponding `dropbox.files.FileMetadata` or `dropbox.files.FolderMetadata` for the item at time of deletion, and not a `dropbox.files.DeletedMetadata` object.

Parameters **path** (*str*) – Path in the user's Dropbox to delete.

Return type `dropbox.files.Metadata`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.files.DeleteError`

files_delete_batch(*entries*)

Delete multiple files/folders at once. This route is asynchronous, which returns a job ID immediately and runs the delete batch asynchronously. Use `files_delete_batch_check()` to check the job status.

Return type `dropbox.files.LaunchEmptyResult`

files_delete_batch_check(*async_job_id*)

Returns the status of an asynchronous job for `files_delete_batch()`. If success, it returns list of result for each entry.

Parameters **async_job_id** (*str*) – Id of the asynchronous job. This is the value of a response returned from the method that launched the job.

Return type `dropbox.files.DeleteBatchJobStatus`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.files.PollError`

files_download(*path*, *rev=None*)

Download a file from a user's Dropbox.

Parameters

- **path** (*str*) – The path of the file to download.
- **rev** (*Nullable*) – Deprecated. Please specify revision in `path` instead.

Return type (`dropbox.files.FileMetadata`, `requests.models.Response`)

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.files.DownloadError`

If you do not consume the entire response body, then you must call close on the response object, otherwise you will max out your available connections. We recommend using the contextlib.closing context manager to ensure this.

files_download_to_file(*download_path*, *path*, *rev=None*)

Download a file from a user's Dropbox.

Parameters

- **download_path** (*str*) – Path on local machine to save file.
- **path** (*str*) – The path of the file to download.
- **rev** (*Nullable*) – Deprecated. Please specify revision in `path` instead.

Return type (`dropbox.files.FileMetadata`, `requests.models.Response`)

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.files.DownloadError`

files_get_metadata(*path*, *include_media_info=False*, *include_deleted=False*, *include_has_explicit_shared_members=False*)

Returns the metadata for a file or folder. Note: Metadata for the root folder is unsupported.

Parameters

- **path** (*str*) – The path of a file or folder on Dropbox.
- **include_media_info** (*bool*) – If true, `FileMetadata.media_info` is set for photo and video.

- **include_deleted** (*bool*) – If true, *dropbox.files.DeletedMetadata* will be returned for deleted file or folder, otherwise `LookupError.not_found` will be returned.
- **include_has_explicit_shared_members** (*bool*) – If true, the results will include a flag for each file indicating whether or not that file has any explicit members.

Return type *dropbox.files.Metadata*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.GetMetadataError*

files_get_preview(*path*, *rev=None*)

Get a preview for a file. Currently previews are only generated for the files with the following extensions: .doc, .docx, .docm, .ppt, .pps, .ppsx, .ppsm, .pptx, .pptm, .xls, .xlsx, .xlsm, .rtf.

Parameters

- **path** (*str*) – The path of the file to preview.
- **rev** (*Nullable*) – Deprecated. Please specify revision in `path` instead.

Return type (*dropbox.files.FileMetadata*, `requests.models.Response`)

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.PreviewError*

If you do not consume the entire response body, then you must call close on the response object, otherwise you will max out your available connections. We recommend using the contextlib.closing context manager to ensure this.

files_get_preview_to_file(*download_path*, *path*, *rev=None*)

Get a preview for a file. Currently previews are only generated for the files with the following extensions: .doc, .docx, .docm, .ppt, .pps, .ppsx, .ppsm, .pptx, .pptm, .xls, .xlsx, .xlsm, .rtf.

Parameters

- **download_path** (*str*) – Path on local machine to save file.
- **path** (*str*) – The path of the file to preview.
- **rev** (*Nullable*) – Deprecated. Please specify revision in `path` instead.

Return type (*dropbox.files.FileMetadata*, `requests.models.Response`)

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.PreviewError*

files_get_temporary_link(*path*)

Get a temporary link to stream content of a file. This link will expire in four hours and afterwards you will get 410 Gone. Content-Type of the link is determined automatically by the file's mime type.

Parameters **path** (*str*) – The path to the file you want a temporary link to.

Return type *dropbox.files.GetTemporaryLinkResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.GetTemporaryLinkError*

files_get_thumbnail(*path*, *format=ThumbnailFormat('jpeg', None)*, *size=ThumbnailSize('w64h64', None)*)

Get a thumbnail for an image. This method currently supports files with the following file extensions: jpg, jpeg, png, tiff, tif, gif and bmp. Photos that are larger than 20MB in size won't be converted to a thumbnail.

Parameters

- **path** (*str*) – The path to the image file you want to thumbnail.
- **format** (*dropbox.files.ThumbnailFormat*) – The format for the thumbnail image, jpeg (default) or png. For images that are photos, jpeg should be preferred, while png is better for screenshots and digital arts.
- **size** (*dropbox.files.ThumbnailSize*) – The size for the thumbnail image.

Return type (*dropbox.files.FileMetadata*, requests.models.Response)

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.ThumbnailError*

If you do not consume the entire response body, then you must call close on the response object, otherwise you will max out your available connections. We recommend using the contextlib.closing context manager to ensure this.

files_get_thumbnail_to_file(*download_path*, *path*, *format=ThumbnailFormat('jpeg', None)*, *size=ThumbnailSize('w64h64', None)*)

Get a thumbnail for an image. This method currently supports files with the following file extensions: jpg, jpeg, png, tiff, tif, gif and bmp. Photos that are larger than 20MB in size won't be converted to a thumbnail.

Parameters

- **download_path** (*str*) – Path on local machine to save file.
- **path** (*str*) – The path to the image file you want to thumbnail.
- **format** (*dropbox.files.ThumbnailFormat*) – The format for the thumbnail image, jpeg (default) or png. For images that are photos, jpeg should be preferred, while png is better for screenshots and digital arts.
- **size** (*dropbox.files.ThumbnailSize*) – The size for the thumbnail image.

Return type (*dropbox.files.FileMetadata*, requests.models.Response)

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.ThumbnailError*

files_list_folder(*path*, *recursive=False*, *include_media_info=False*, *include_deleted=False*, *include_has_explicit_shared_members=False*)

Starts returning the contents of a folder. If the result's `ListFolderResult.has_more` field is `True`, call *files_list_folder_continue()* with the returned `ListFolderResult.cursor` to retrieve more entries. If you're using `ListFolderArg.recursive` set to `True` to keep a local cache of the contents of a Dropbox account, iterate through each entry in order and process them as follows to keep your local state in sync: For each *dropbox.files.FileMetadata*, store the new entry at the given path in your local state. If the required parent folders don't exist yet, create them. If there's already something else at the given path, replace it and remove all its children. For each *dropbox.files.FolderMetadata*, store the new entry at the given path in your local state. If the required parent folders don't exist yet, create them. If there's already something else at the given path, replace it but leave the children as they are. Check the new entry's `FolderSharingInfo.read_only` and set all its children's read-only statuses to match. For each *dropbox.files.DeletedMetadata*,

if your local state has something at the given path, remove it and all its children. If there's nothing at the given path, ignore this entry.

Parameters

- **path** (*str*) – The path to the folder you want to see the contents of.
- **recursive** (*bool*) – If true, the list folder operation will be applied recursively to all subfolders and the response will contain contents of all subfolders.
- **include_media_info** (*bool*) – If true, `FileMetadata.media_info` is set for photo and video.
- **include_deleted** (*bool*) – If true, the results will include entries for files and folders that used to exist but were deleted.
- **include_has_explicit_shared_members** (*bool*) – If true, the results will include a flag for each file indicating whether or not that file has any explicit members.

Return type *dropbox.files.ListFolderResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.ListFolderError*

files_list_folder_continue(*cursor*)

Once a cursor has been retrieved from *files_list_folder()*, use this to paginate through all files and retrieve updates to the folder, following the same rules as documented for *files_list_folder()*.

Parameters **cursor** (*str*) – The cursor returned by your last call to *files_list_folder()* or *files_list_folder_continue()*.

Return type *dropbox.files.ListFolderResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.ListFolderContinueError*

files_list_folder_get_latest_cursor(*path*, *recursive=False*, *include_media_info=False*, *include_deleted=False*, *include_has_explicit_shared_members=False*)

A way to quickly get a cursor for the folder's state. Unlike *files_list_folder()*, *files_list_folder_get_latest_cursor()* doesn't return any entries. This endpoint is for app which only needs to know about new files and modifications and doesn't need to know about files that already exist in Dropbox.

Parameters

- **path** (*str*) – The path to the folder you want to see the contents of.
- **recursive** (*bool*) – If true, the list folder operation will be applied recursively to all subfolders and the response will contain contents of all subfolders.
- **include_media_info** (*bool*) – If true, `FileMetadata.media_info` is set for photo and video.
- **include_deleted** (*bool*) – If true, the results will include entries for files and folders that used to exist but were deleted.
- **include_has_explicit_shared_members** (*bool*) – If true, the results will include a flag for each file indicating whether or not that file has any explicit members.

Return type *dropbox.files.ListFolderGetLatestCursorResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.ListFolderError*

files_list_folder_longpoll(*cursor*, *timeout=30*)

A longpoll endpoint to wait for changes on an account. In conjunction with *files_list_folder_continue()*, this call gives you a low-latency way to monitor an account for file changes. The connection will block until there are changes available or a timeout occurs. This endpoint is useful mostly for client-side apps. If you're looking for server-side notifications, check out our webhooks documentation.

Parameters

- **cursor** (*str*) – A cursor as returned by *files_list_folder()* or *files_list_folder_continue()*. Cursors retrieved by setting `ListFolderArg.include_media_info` to `True` are not supported.
- **timeout** (*long*) – A timeout in seconds. The request will block for at most this length of time, plus up to 90 seconds of random jitter added to avoid the thundering herd problem. Care should be taken when using this parameter, as some network infrastructure does not support long timeouts.

Return type *dropbox.files.ListFolderLongpollResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.ListFolderLongpollError*

files_list_revisions(*path*, *limit=10*)

Return revisions of a file.

Parameters

- **path** (*str*) – The path to the file you want to see the revisions of.
- **limit** (*long*) – The maximum number of revision entries returned.

Return type *dropbox.files.ListRevisionsResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.ListRevisionsError*

files_move(*from_path*, *to_path*, *allow_shared_folder=False*, *autorename=False*)

Move a file or folder to a different location in the user's Dropbox. If the source path is a folder all its contents will be moved.

Parameters

- **allow_shared_folder** (*bool*) – If true, *files_copy()* will copy contents in shared folder, otherwise `RelocationError.cant_copy_shared_folder` will be returned if `from_path` contains shared folder. This field is always true for *files_move()*.
- **autorename** (*bool*) – If there's a conflict, have the Dropbox server try to autorename the file to avoid the conflict.

Return type *dropbox.files.Metadata*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: `dropbox.files.RelocationError`

files_move_batch(*entries*, *allow_shared_folder=False*, *autorename=False*)

Move multiple files or folders to different locations at once in the user's Dropbox. This route is 'all or nothing', which means if one entry fails, the whole transaction will abort. This route will return job ID immediately and do the async moving job in background. Please use `files_move_batch_check()` to check the job status.

Parameters

- **entries** (*list*) – List of entries to be moved or copied. Each entry is `dropbox.files.RelocationPath`.
- **allow_shared_folder** (*bool*) – If true, `files_copy_batch()` will copy contents in shared folder, otherwise `RelocationError.cant_copy_shared_folder` will be returned if `RelocationPath.from_path` contains shared folder. This field is always true for `files_move_batch()`.
- **autorename** (*bool*) – If there's a conflict with any file, have the Dropbox server try to autorename that file to avoid the conflict.

Return type `dropbox.files.LaunchEmptyResult`

files_move_batch_check(*async_job_id*)

Returns the status of an asynchronous job for `files_move_batch()`. If success, it returns list of results for each entry.

Parameters **async_job_id** (*str*) – Id of the asynchronous job. This is the value of a response returned from the method that launched the job.

Return type `dropbox.files.RelocationBatchJobStatus`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.files.PollError`

files_permanently_delete(*path*)

Permanently delete the file or folder at a given path (see https://www.dropbox.com/en/help/40). Note: This endpoint is only available for Dropbox Business apps.

Parameters **path** (*str*) – Path in the user's Dropbox to delete.

Return type None

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.files.DeleteError`

files_properties_add(*path*, *property_groups*)

Add custom properties to a file using a filled property template. See properties/template/add to create new property templates.

Parameters

- **path** (*str*) – A unique identifier for the file.
- **property_groups** (*list*) – Filled custom property templates associated with a file.

Return type None

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: *dropbox.files.AddPropertiesError*

files_properties_overwrite(*path*, *property_groups*)

Overwrite custom properties from a specified template associated with a file.

Parameters

- **path** (*str*) – A unique identifier for the file.
- **property_groups** (*list*) – Filled custom property templates associated with a file.

Return type None

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.InvalidPropertyGroupError*

files_properties_remove(*path*, *property_template_ids*)

Remove all custom properties from a specified template associated with a file. To remove specific property key value pairs, see *files_properties_update()*. To update a property template, see properties/template/update. Property templates can't be removed once created.

Parameters

- **path** (*str*) – A unique identifier for the file.
- **property_template_ids** (*list*) – A list of identifiers for a property template created by route properties/template/add.

Return type None

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.RemovePropertiesError*

files_properties_template_get(*template_id*)

Get the schema for a specified template.

Parameters **template_id** (*str*) – An identifier for property template added by route properties/template/add.

Return type dropbox.files.GetPropertyTemplateResult

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: dropbox.files.PropertyTemplateError

files_properties_template_list()

Get the property template identifiers for a user. To get the schema of each template use *files_properties_template_get()*.

Return type dropbox.files.ListPropertyTemplateIds

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: dropbox.files.PropertyTemplateError

files_properties_update(*path*, *update_property_groups*)

Add, update or remove custom properties from a specified template associated with a file. Fields that already exist and not described in the request will not be modified.

Parameters

- **path** (*str*) – A unique identifier for the file.
- **update_property_groups** (*list*) – Filled custom property templates associated with a file.

Return type None

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.UpdatePropertiesError*

files_restore (*path*, *rev*)

Restore a file to a specific revision.

Parameters

- **path** (*str*) – The path to the file you want to restore.
- **rev** (*str*) – The revision to restore for the file.

Return type *dropbox.files.FileMetadata*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.RestoreError*

files_save_url (*path*, *url*)

Save a specified URL into a file in user's Dropbox. If the given path already exists, the file will be renamed to avoid the conflict (e.g. myfile (1).txt).

Parameters

- **path** (*str*) – The path in Dropbox where the URL will be saved to.
- **url** (*str*) – The URL to be saved.

Return type *dropbox.files.SaveUrlResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.SaveUrlError*

files_save_url_check_job_status (*async_job_id*)

Check the status of a *files_save_url()* job.

Parameters **async_job_id** (*str*) – Id of the asynchronous job. This is the value of a response returned from the method that launched the job.

Return type *dropbox.files.SaveUrlJobStatus*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: `dropbox.files.PollError`

files_search (*path*, *query*, *start=0*, *max_results=100*, *mode=SearchMode('filename'*, *None)*)

Searches for files and folders. Note: Recent changes may not immediately be reflected in search results due to a short delay in indexing.

Parameters

- **path** (*str*) – The path in the user's Dropbox to search. Should probably be a folder.

- **query** (*str*) – The string to search for. The search string is split on spaces into multiple tokens. For file name searching, the last token is used for prefix matching (i.e. "bat c" matches "bat cave" but not "batman car").
- **start** (*long*) – The starting index within the search results (used for paging).
- **max_results** (*long*) – The maximum number of search results to return.
- **mode** (*dropbox.files.SearchMode*) – The search mode (filename, filename_and_content, or deleted_filename). Note that searching file content is only available for Dropbox Business accounts.

Return type *dropbox.files.SearchResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.SearchError*

files_upload(*f*, *path*, *mode=WriteMode('add', None)*, *autorename=False*, *client_modified=None*, *mute=False*)

Create a new file with the contents provided in the request. Do not use this to upload a file larger than 150 MB. Instead, create an upload session with *files_upload_session_start()*.

Parameters

- **f** (*bytes*) – Contents to upload.
- **path** (*str*) – Path in the user's Dropbox to save the file.
- **mode** (*dropbox.files.WriteMode*) – Selects what to do if the file already exists.
- **autorename** (*bool*) – If there's a conflict, as determined by `mode`, have the Dropbox server try to autorename the file to avoid conflict.
- **client_modified** (*Nullable*) – The value to store as the `client_modified` timestamp. Dropbox automatically records the time at which the file was written to the Dropbox servers. It can also record an additional timestamp, provided by Dropbox desktop clients, mobile clients, and API apps of when the file was actually created or modified.
- **mute** (*bool*) – Normally, users are made aware of any file modifications in their Dropbox account via notifications in the client software. If `True`, this tells the clients that this modification shouldn't result in a user notification.

Return type *dropbox.files.FileMetadata*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.files.UploadError*

files_upload_session_append(*f*, *session_id*, *offset*)

Append more data to an upload session. A single request should not upload more than 150 MB of file contents.

Parameters

- **f** (*bytes*) – Contents to upload.
- **session_id** (*str*) – The upload session ID (returned by *files_upload_session_start()*).
- **offset** (*long*) – The amount of data that has been uploaded so far. We use this to make sure upload data isn't lost or duplicated in the event of a network error.

Return type None

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.files.UploadSessionLookupError`

files_upload_session_append_v2(*f*, *cursor*, *close=False*)

Append more data to an upload session. When the parameter close is set, this call will close the session. A single request should not upload more than 150 MB of file contents.

Parameters

- **f** (*bytes*) – Contents to upload.
- **cursor** (`dropbox.files.UploadSessionCursor`) – Contains the upload session ID and the offset.
- **close** (*bool*) – If true, the current session will be closed, at which point you won't be able to call `files_upload_session_append_v2()` anymore with the current session.

Return type None

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.files.UploadSessionLookupError`

files_upload_session_finish(*f*, *cursor*, *commit*)

Finish an upload session and save the uploaded data to the given file path. A single request should not upload more than 150 MB of file contents.

Parameters

- **f** (*bytes*) – Contents to upload.
- **cursor** (`dropbox.files.UploadSessionCursor`) – Contains the upload session ID and the offset.
- **commit** (`dropbox.files.CommitInfo`) – Contains the path and other optional modifiers for the commit.

Return type `dropbox.files.FileMetadata`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.files.UploadSessionFinishError`

files_upload_session_finish_batch(*entries*)

This route helps you commit many files at once into a user's Dropbox. Use `files_upload_session_start()` and `files_upload_session_append_v2()` to upload file contents. We recommend uploading many files in parallel to increase throughput. Once the file contents have been uploaded, rather than calling `files_upload_session_finish()`, use this route to finish all your upload sessions in a single request. `UploadSessionStartArg.close` or `UploadSessionAppendArg.close` needs to be true for the last `files_upload_session_start()` or `files_upload_session_append_v2()` call. This route will return a job_id immediately and do the async commit job in background. Use `files_upload_session_finish_batch_check()` to check the job status. For the same account, this route should be executed serially. That means you should not start the next job before current job finishes. We allow up to 1000 entries in a single request.

Parameters **entries** (*list*) – Commit information for each file in the batch.

Return type `dropbox.files.LaunchEmptyResult`

files_upload_session_finish_batch_check(*async_job_id*)

Returns the status of an asynchronous job for `files_upload_session_finish_batch()`. If success, it returns list of result for each entry.

Parameters **async_job_id** (*str*) – Id of the asynchronous job. This is the value of a response returned from the method that launched the job.

Return type `dropbox.files.UploadSessionFinishBatchJobStatus`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.files.PollError`

files_upload_session_start(*f*, *close=False*)

Upload sessions allow you to upload a single file in one or more requests, for example where the size of the file is greater than 150 MB. This call starts a new upload session with the given data. You can then use `files_upload_session_append_v2()` to add more data and `files_upload_session_finish()` to save all the data to a file in Dropbox. A single request should not upload more than 150 MB of file contents.

Parameters

- **f** (*bytes*) – Contents to upload.
- **close** (*bool*) – If true, the current session will be closed, at which point you won't be able to call `files_upload_session_append_v2()` anymore with the current session.

Return type `dropbox.files.UploadSessionStartResult`

request(*route*, *namespace*, *request_arg*, *request_binary*, *timeout=None*)

Makes a request to the Dropbox API and in the process validates that the route argument and result are the expected data types. The request_arg is converted to JSON based on the arg_data_type. Likewise, the response is deserialized from JSON and converted to an object based on the {result,error}_data_type.

Parameters

- **host** – The Dropbox API host to connect to.
- **route** (`datatypes.stone_base.Route`) – The route to make the request to.
- **request_arg** – Argument for the route that conforms to the validator specified by route.arg_type.
- **request_binary** – String or file pointer representing the binary payload. Use None if there is no binary payload.
- **timeout** (*Optional[float]*) – Maximum duration in seconds that client will wait for any single packet from the server. After the timeout the client will give up on connection. If *None*, will use default timeout set on Dropbox object. Defaults to *None*.

Returns The route's result.

request_json_object(*host*, *route_name*, *route_style*, *request_arg*, *request_binary*, *timeout=None*)

Makes a request to the Dropbox API, taking a JSON-serializable Python object as an argument, and returning one as a response.

Parameters

- **host** – The Dropbox API host to connect to.
- **route_name** – The name of the route to invoke.
- **route_style** – The style of the route.

- **request_arg** (*str*) – A JSON-serializable Python object representing the argument for the route.
- **request_binary** (*Optional[bytes]*) – Bytes representing the binary payload. Use None if there is no binary payload.
- **timeout** (*Optional[float]*) – Maximum duration in seconds that client will wait for any single packet from the server. After the timeout the client will give up on connection. If *None*, will use default timeout set on Dropbox object. Defaults to *None*.

Returns The route's result as a JSON-serializable Python object.

request_json_string(*host*, *func_name*, *route_style*, *request_json_arg*, *request_binary*, *timeout=None*)

See *request_json_string_with_retry()* for description of parameters.

request_json_string_with_retry(*host*, *route_name*, *route_style*, *request_json_arg*, *request_binary*, *timeout=None*)

See *request_json_object()* for description of parameters.

Parameters **request_json_arg** – A string representing the serialized JSON argument to the route.

sharing_add_file_member(*file*, *members*, *custom_message=None*, *quiet=False*, *access_level=AccessLevel('viewer', None)*, *add_message_as_comment=False*)

Adds specified members to a file.

Parameters

- **file** (*str*) – File to which to add members.
- **members** (*list*) – Members to add. Note that even an email address is given, this may result in a user being directy added to the membership if that email is the user's main account email.
- **custom_message** (*Nullable*) – Message to send to added members in their invitation.
- **quiet** (*bool*) – Whether added members should be notified via device notifications of their invitation.
- **access_level** (*dropbox.sharing.AccessLevel*) – AccessLevel union object, describing what access level we want to give new members.
- **add_message_as_comment** (*bool*) – If the custom message should be added as a comment on the file.

Return type list

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.AddFileMemberError*

sharing_add_folder_member(*shared_folder_id*, *members*, *quiet=False*, *custom_message=None*)

Allows an owner or editor (if the ACL update policy allows) of a shared folder to add another member. For the new member to get access to all the functionality for this folder, you will need to call *sharing_mount_folder()* on their behalf. Apps must have full Dropbox access to use this endpoint.

Parameters

- **shared_folder_id** (*str*) – The ID for the shared folder.

- **members** (*list*) – The intended list of members to add. Added members will receive invites to join the shared folder.
- **quiet** (*bool*) – Whether added members should be notified via email and device notifications of their invite.
- **custom_message** (*Nullable*) – Optional message to display to added members in their invitation.

Return type None

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.AddFolderMemberError*

sharing_change_file_member_access (*file*, *member*, *access_level*)

Changes a member's access on a shared file.

Parameters

- **file** (*str*) – File for which we are changing a member's access.
- **member** (*dropbox.sharing.MemberSelector*) – The member whose access we are changing.
- **access_level** (*dropbox.sharing.AccessLevel*) – The new access level for the member.

Return type *dropbox.sharing.FileMemberActionResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.FileMemberActionError*

sharing_check_job_status (*async_job_id*)

Returns the status of an asynchronous job. Apps must have full Dropbox access to use this endpoint.

Parameters **async_job_id** (*str*) – Id of the asynchronous job. This is the value of a response returned from the method that launched the job.

Return type *dropbox.sharing.JobStatus*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: `dropbox.sharing.PollError`

sharing_check_remove_member_job_status (*async_job_id*)

Returns the status of an asynchronous job for sharing a folder. Apps must have full Dropbox access to use this endpoint.

Parameters **async_job_id** (*str*) – Id of the asynchronous job. This is the value of a response returned from the method that launched the job.

Return type *dropbox.sharing.RemoveMemberJobStatus*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: `dropbox.sharing.PollError`

sharing_check_share_job_status (*async_job_id*)

Returns the status of an asynchronous job for sharing a folder. Apps must have full Dropbox access to use this endpoint.

Parameters **async_job_id** (*str*) – Id of the asynchronous job. This is the value of a response returned from the method that launched the job.

Return type *dropbox.sharing.ShareFolderJobStatus*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: `dropbox.sharing.PollError`

sharing_create_shared_link (*path*, *short_url=False*, *pending_upload=None*)

Create a shared link. If a shared link already exists for the given path, that link is returned. Note that in the returned *dropbox.sharing.PathLinkMetadata*, the `PathLinkMetadata.url` field is the shortened URL if `CreateSharedLinkArg.short_url` argument is set to `True`. Previously, it was technically possible to break a shared link by moving or renaming the corresponding file or folder. In the future, this will no longer be the case, so your app shouldn't rely on this behavior. Instead, if your app needs to revoke a shared link, use *sharing_revoke_shared_link()*.

Parameters

- **path** (*str*) – The path to share.
- **short_url** (*bool*) – Whether to return a shortened URL.
- **pending_upload** (*Nullable*) – If it's okay to share a path that does not yet exist, set this to either `PendingUploadMode.file` or `PendingUploadMode.folder` to indicate whether to assume it's a file or folder.

Return type *dropbox.sharing.PathLinkMetadata*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.CreateSharedLinkError*

sharing_create_shared_link_with_settings (*path*, *settings=None*)

Create a shared link with custom settings. If no settings are given then the default visibility is `RequestedVisibility.public` (The resolved visibility, though, may depend on other aspects such as team and shared folder settings).

Parameters

- **path** (*str*) – The path to be shared by the shared link
- **settings** (*Nullable*) – The requested settings for the newly created shared link

Return type *dropbox.sharing.SharedLinkMetadata*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.CreateSharedLinkWithSettingsError*

sharing_get_file_metadata (*file*, *actions=None*)

Returns shared file metadata.

Parameters

- **file** (*str*) – The file to query.
- **actions** (*Nullable*) – File actions to query.

Return type *dropbox.sharing.SharedFileMetadata*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: `dropbox.sharing.GetFileMetadataError`

sharing_get_file_metadata_batch(*files*, *actions=None*)

Returns shared file metadata.

Parameters

- **files** (*list*) – The files to query.
- **actions** (*Nullable*) – File actions to query.

Return type list

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.sharing.SharingUserError`

sharing_get_folder_metadata(*shared_folder_id*, *actions=None*)

Returns shared folder metadata by its folder ID. Apps must have full Dropbox access to use this endpoint.

Parameters

- **shared_folder_id** (*str*) – The ID for the shared folder.
- **actions** (*Nullable*) – This is a list indicating whether the returned folder data will include a boolean value `FolderPermission.allow` that describes whether the current user can perform the FolderAction on the folder.

Return type `dropbox.sharing.SharedFolderMetadata`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.sharing.SharedFolderAccessError`

sharing_get_shared_link_file(*url*, *path=None*, *link_password=None*)

Download the shared link's file from a user's Dropbox.

Parameters

- **url** (*str*) – URL of the shared link.
- **path** (*Nullable*) – If the shared link is to a folder, this parameter can be used to retrieve the metadata for a specific file or sub-folder in this folder. A relative path should be used.
- **link_password** (*Nullable*) – If the shared link has a password, this parameter can be used.

Return type (`dropbox.sharing.SharedLinkMetadata`, `requests.models.Response`)

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.sharing.GetSharedLinkFileError`

If you do not consume the entire response body, then you must call close on the response object, otherwise you will max out your available connections. We recommend using the contextlib.closing context manager to ensure this.

sharing_get_shared_link_file_to_file(*download_path*, *url*, *path=None*, *link_password=None*)

Download the shared link's file from a user's Dropbox.

Parameters

- **download_path** (*str*) – Path on local machine to save file.
- **url** (*str*) – URL of the shared link.
- **path** (*Nullable*) – If the shared link is to a folder, this parameter can be used to retrieve the metadata for a specific file or sub-folder in this folder. A relative path should be used.
- **link_password** (*Nullable*) – If the shared link has a password, this parameter can be used.

Return type (*dropbox.sharing.SharedLinkMetadata*, requests.models.Response)

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.GetSharedLinkFileError*

sharing_get_shared_link_metadata(*url*, *path=None*, *link_password=None*)

Get the shared link's metadata.

Parameters

- **url** (*str*) – URL of the shared link.
- **path** (*Nullable*) – If the shared link is to a folder, this parameter can be used to retrieve the metadata for a specific file or sub-folder in this folder. A relative path should be used.
- **link_password** (*Nullable*) – If the shared link has a password, this parameter can be used.

Return type *dropbox.sharing.SharedLinkMetadata*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.SharedLinkError*

sharing_get_shared_links(*path=None*)

Returns a list of *dropbox.sharing.LinkMetadata* objects for this user, including collection links. If no path is given, returns a list of all shared links for the current user, including collection links. If a non-empty path is given, returns a list of all shared links that allow access to the given path. Collection links are never returned in this case. Note that the url field in the response is never the shortened URL.

Parameters **path** (*Nullable*) – See *sharing_get_shared_links()* description.

Return type *dropbox.sharing.GetSharedLinksResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.GetSharedLinksError*

sharing_list_file_members(*file*, *actions=None*, *include_inherited=True*, *limit=100*)

Use to obtain the members who have been invited to a file, both inherited and uninherited members.

Parameters

- **file** (*str*) – The file for which you want to see members.
- **actions** (*Nullable*) – The actions for which to return permissions on a member
- **include_inherited** (*bool*) – Whether to include members who only have access from a parent shared folder.

- **limit** (*long*) – Number of members to return max per query. Defaults to 100 if no limit is specified.

Return type *dropbox.sharing.SharedFileMembers*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.ListFileMembersError*

sharing_list_file_members_batch(*files*, *limit=10*)

Get members of multiple files at once. The arguments to this route are more limited, and the limit on query result size per file is more strict. To customize the results more, use the individual file endpoint. Inherited users are not included in the result, and permissions are not returned for this endpoint.

Parameters

- **files** (*list*) – Files for which to return members.
- **limit** (*long*) – Number of members to return max per query. Defaults to 10 if no limit is specified.

Return type list

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.SharingUserError*

sharing_list_file_members_continue(*cursor*)

Once a cursor has been retrieved from *sharing_list_file_members()* or *sharing_list_file_members_batch()*, use this to paginate through all shared file members.

Parameters **cursor** (*str*) – The cursor returned by your last call to *sharing_list_file_members()*, *sharing_list_file_members_continue()*, or *sharing_list_file_members_batch()*.

Return type *dropbox.sharing.SharedFileMembers*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.ListFileMembersContinueError*

sharing_list_folder_members(*shared_folder_id*, *actions=None*, *limit=1000*)

Returns shared folder membership by its folder ID. Apps must have full Dropbox access to use this endpoint.

Parameters **shared_folder_id** (*str*) – The ID for the shared folder.

Return type *dropbox.sharing.SharedFolderMembers*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.SharedFolderAccessError*

sharing_list_folder_members_continue(*cursor*)

Once a cursor has been retrieved from *sharing_list_folder_members()*, use this to paginate through all shared folder members. Apps must have full Dropbox access to use this endpoint.

Parameters **cursor** (*str*) – The cursor returned by your last call to *sharing_list_folder_members()* or *sharing_list_folder_members_continue()*.

Return type *dropbox.sharing.SharedFolderMembers*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.ListFolderMembersContinueError*

sharing_list_folders(*limit=1000*, *actions=None*)

Return the list of all shared folders the current user has access to. Apps must have full Dropbox access to use this endpoint.

Parameters

- **limit** (*long*) – The maximum number of results to return per request.
- **actions** (*Nullable*) – This is a list indicating whether each returned folder data entry will include a boolean field `FolderPermission.allow` that describes whether the current user can perform the *FolderAction* on the folder.

Return type *dropbox.sharing.ListFoldersResult*

sharing_list_folders_continue(*cursor*)

Once a cursor has been retrieved from *sharing_list_folders()*, use this to paginate through all shared folders. The cursor must come from a previous call to *sharing_list_folders()* or *sharing_list_folders_continue()*. Apps must have full Dropbox access to use this endpoint.

Parameters **cursor** (*str*) – The cursor returned by the previous API call specified in the endpoint description.

Return type *dropbox.sharing.ListFoldersResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.ListFoldersContinueError*

sharing_list_mountable_folders(*limit=1000*, *actions=None*)

Return the list of all shared folders the current user can mount or unmount. Apps must have full Dropbox access to use this endpoint.

Parameters

- **limit** (*long*) – The maximum number of results to return per request.
- **actions** (*Nullable*) – This is a list indicating whether each returned folder data entry will include a boolean field `FolderPermission.allow` that describes whether the current user can perform the *FolderAction* on the folder.

Return type *dropbox.sharing.ListFoldersResult*

sharing_list_mountable_folders_continue(*cursor*)

Once a cursor has been retrieved from *sharing_list_mountable_folders()*, use this to paginate through all mountable shared folders. The cursor must come from a previous call to *sharing_list_mountable_folders()* or *sharing_list_mountable_folders_continue()*. Apps must have full Dropbox access to use this endpoint.

Parameters **cursor** (*str*) – The cursor returned by the previous API call specified in the endpoint description.

Return type *dropbox.sharing.ListFoldersResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.ListFoldersContinueError*

sharing_list_received_files(*limit=100*, *actions=None*)

Returns a list of all files shared with current user. Does not include files the user has received via shared folders, and does not include unclaimed invitations.

Parameters

- **limit** (*long*) – Number of files to return max per query. Defaults to 100 if no limit is specified.
- **actions** (*Nullable*) – File actions to query.

Return type *dropbox.sharing.ListFilesResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.SharingUserError*

sharing_list_received_files_continue(*cursor*)

Get more results with a cursor from *sharing_list_received_files()*.

Parameters **cursor** (*str*) – Cursor in `ListFilesResult.cursor`

Return type *dropbox.sharing.ListFilesResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.ListFilesContinueError*

sharing_list_shared_links(*path=None*, *cursor=None*, *direct_only=None*)

List shared links of this user. If no path is given, returns a list of all shared links for the current user. If a non-empty path is given, returns a list of all shared links that allow access to the given path - direct links to the given path and links to parent folders of the given path. Links to parent folders can be suppressed by setting direct_only to true.

Parameters

- **path** (*Nullable*) – See *sharing_list_shared_links()* description.
- **cursor** (*Nullable*) – The cursor returned by your last call to *sharing_list_shared_links()*.
- **direct_only** (*Nullable*) – See *sharing_list_shared_links()* description.

Return type *dropbox.sharing.ListSharedLinksResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.ListSharedLinksError*

sharing_modify_shared_link_settings(*url*, *settings*, *remove_expiration=False*)

Modify the shared link's settings. If the requested visibility conflict with the shared links policy of the team or the shared folder (in case the linked file is part of a shared folder) then the `LinkPermissions.resolved_visibility` of the returned *dropbox.sharing.SharedLinkMetadata* will reflect the actual visibility of the shared link and the `LinkPermissions.requested_visibility` will reflect the requested visibility.

Parameters

- **url** (*str*) – URL of the shared link to change its settings

- **settings** (*dropbox.sharing.SharedLinkSettings*) – Set of settings for the shared link.
- **remove_expiration** (*bool*) – If set to true, removes the expiration of the shared link.

Return type *dropbox.sharing.SharedLinkMetadata*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.ModifySharedLinkSettingsError*

sharing_mount_folder (*shared_folder_id*)

The current user mounts the designated folder. Mount a shared folder for a user after they have been added as a member. Once mounted, the shared folder will appear in their Dropbox. Apps must have full Dropbox access to use this endpoint.

Parameters **shared_folder_id** (*str*) – The ID of the shared folder to mount.

Return type *dropbox.sharing.SharedFolderMetadata*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.MountFolderError*

sharing_relinquish_file_membership (*file*)

The current user relinquishes their membership in the designated file. Note that the current user may still have inherited access to this file through the parent folder. Apps must have full Dropbox access to use this endpoint.

Parameters **file** (*str*) – The path or id for the file.

Return type None

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.RelinquishFileMembershipError*

sharing_relinquish_folder_membership (*shared_folder_id*, *leave_a_copy=False*)

The current user relinquishes their membership in the designated shared folder and will no longer have access to the folder. A folder owner cannot relinquish membership in their own folder. This will run synchronously if leave_a_copy is false, and asynchronously if leave_a_copy is true. Apps must have full Dropbox access to use this endpoint.

Parameters

- **shared_folder_id** (*str*) – The ID for the shared folder.
- **leave_a_copy** (*bool*) – Keep a copy of the folder's contents upon relinquishing membership.

Return type dropbox.sharing.LaunchEmptyResult

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.RelinquishFolderMembershipError*

sharing_remove_file_member (*file*, *member*)

Identical to remove_file_member_2 but with less information returned.

Parameters

- **file** (*str*) – File from which to remove members.
- **member** (*dropbox.sharing.MemberSelector*) – Member to remove from this file. Note that even if an email is specified, it may result in the removal of a user (not an invitee) if the user's main account corresponds to that email address.

Return type *dropbox.sharing.FileMemberActionIndividualResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.RemoveFileMemberError*

sharing_remove_file_member_2 (*file*, *member*)

Removes a specified member from the file.

Parameters

- **file** (*str*) – File from which to remove members.
- **member** (*dropbox.sharing.MemberSelector*) – Member to remove from this file. Note that even if an email is specified, it may result in the removal of a user (not an invitee) if the user's main account corresponds to that email address.

Return type *dropbox.sharing.FileMemberRemoveActionResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.RemoveFileMemberError*

sharing_remove_folder_member (*shared_folder_id*, *member*, *leave_a_copy*)

Allows an owner or editor (if the ACL update policy allows) of a shared folder to remove another member. Apps must have full Dropbox access to use this endpoint.

Parameters

- **shared_folder_id** (*str*) – The ID for the shared folder.
- **member** (*dropbox.sharing.MemberSelector*) – The member to remove from the folder.
- **leave_a_copy** (*bool*) – If true, the removed user will keep their copy of the folder after it's unshared, assuming it was mounted. Otherwise, it will be removed from their Dropbox. Also, this must be set to false when kicking a group.

Return type `dropbox.sharing.LaunchResultBase`

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.RemoveFolderMemberError*

sharing_revoke_shared_link (*url*)

Revoke a shared link. Note that even after revoking a shared link to a file, the file may be accessible if there are shared links leading to any of the file parent folders. To list all shared links that enable access to a specific file, you can use the *sharing_list_shared_links()* with the file as the `ListSharedLinksArg.path` argument.

Parameters **url** (*str*) – URL of the shared link.

Return type None

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: `dropbox.sharing.RevokeSharedLinkError`

sharing_share_folder(*path*, *member_policy=MemberPolicy('anyone', None)*, *acl_update_policy=AclUpdatePolicy('owner', None)*, *shared_link_policy=SharedLinkPolicy('anyone', None)*, *force_async=False*)

Share a folder with collaborators. Most sharing will be completed synchronously. Large folders will be completed asynchronously. To make testing the async case repeatable, set *ShareFolderArg.force_async*. If a `ShareFolderLaunch.async_job_id` is returned, you'll need to call `sharing_check_share_job_status()` until the action completes to get the metadata for the folder. Apps must have full Dropbox access to use this endpoint.

Parameters

- **path** (*str*) – The path to the folder to share. If it does not exist, then a new one is created.
- **member_policy** (`dropbox.sharing.MemberPolicy`) – Who can be a member of this shared folder. Only applicable if the current user is on a team.
- **acl_update_policy** (`dropbox.sharing.AclUpdatePolicy`) – Who can add and remove members of this shared folder.
- **shared_link_policy** (`dropbox.sharing.SharedLinkPolicy`) – The policy to apply to shared links created for content inside this shared folder. The current user must be on a team to set this policy to `SharedLinkPolicy.members`.
- **force_async** (*bool*) – Whether to force the share to happen asynchronously.

Return type `dropbox.sharing.ShareFolderLaunch`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.sharing.ShareFolderError`

sharing_transfer_folder(*shared_folder_id*, *to_dropbox_id*)

Transfer ownership of a shared folder to a member of the shared folder. User must have `AccessLevel.owner` access to the shared folder to perform a transfer. Apps must have full Dropbox access to use this endpoint.

Parameters

- **shared_folder_id** (*str*) – The ID for the shared folder.
- **to_dropbox_id** (*str*) – A account or team member ID to transfer ownership to.

Return type None

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.sharing.TransferFolderError`

sharing_unmount_folder(*shared_folder_id*)

The current user unmounts the designated folder. They can re-mount the folder at a later time using `sharing_mount_folder()`. Apps must have full Dropbox access to use this endpoint.

Parameters **shared_folder_id** (*str*) – The ID for the shared folder.

Return type None

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: *dropbox.sharing.UnmountFolderError*

sharing_unshare_file(*file*)

Remove all members from this file. Does not remove inherited members.

Parameters **file** (*str*) – The file to unshare.

Return type None

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.UnshareFileError*

sharing_unshare_folder(*shared_folder_id*, *leave_a_copy=False*)

Allows a shared folder owner to unshare the folder. You'll need to call *sharing_check_job_status()* to determine if the action has completed successfully. Apps must have full Dropbox access to use this endpoint.

Parameters

- **shared_folder_id** (*str*) – The ID for the shared folder.
- **leave_a_copy** (*bool*) – If true, members of this shared folder will get a copy of this folder after it's unshared. Otherwise, it will be removed from their Dropbox. The current user, who is an owner, will always retain their copy.

Return type dropbox.sharing.LaunchEmptyResult

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.UnshareFolderError*

sharing_update_folder_member(*shared_folder_id*, *member*, *access_level*)

Allows an owner or editor of a shared folder to update another member's permissions. Apps must have full Dropbox access to use this endpoint.

Parameters

- **shared_folder_id** (*str*) – The ID for the shared folder.
- **member** (*dropbox.sharing.MemberSelector*) – The member of the shared folder to update. Only the MemberSelector.dropbox_id may be set at this time.
- **access_level** (*dropbox.sharing.AccessLevel*) – The new access level for member. AccessLevel.owner is disallowed.

Return type *dropbox.sharing.MemberAccessLevelResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.UpdateFolderMemberError*

sharing_update_folder_policy(*shared_folder_id*, *member_policy=None*, *acl_update_policy=None*, *shared_link_policy=None*)

Update the sharing policies for a shared folder. User must have AccessLevel.owner access to the shared folder to update its policies. Apps must have full Dropbox access to use this endpoint.

Parameters

- **shared_folder_id** (*str*) – The ID for the shared folder.
- **member_policy** (*Nullable*) – Who can be a member of this shared folder. Only applicable if the current user is on a team.

- **acl_update_policy** (*Nullable*) – Who can add and remove members of this shared folder.
- **shared_link_policy** (*Nullable*) – The policy to apply to shared links created for content inside this shared folder. The current user must be on a team to set this policy to `SharedLinkPolicy.members`.

Return type *dropbox.sharing.SharedFolderMetadata*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.sharing.UpdateFolderPolicyError*

users_get_account(*account_id*)

Get information about a user's account.

Parameters **account_id** (*str*) – A user's account identifier.

Return type *dropbox.users.BasicAccount*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.users.GetAccountError*

users_get_account_batch(*account_ids*)

Get information about multiple user accounts. At most 300 accounts may be queried per request.

Parameters **account_ids** (*list*) – List of user account identifiers. Should not contain any duplicate account IDs.

Return type list

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.users.GetAccountBatchError*

users_get_current_account()

Get information about the current user's account.

Return type *dropbox.users.FullAccount*

users_get_space_usage()

Get the space usage information for the current user's account.

Return type *dropbox.users.SpaceUsage*

class dropbox.dropbox.**DropboxTeam**(*oauth2_access_token*, *max_retries_on_error=4*, *max_retries_on_rate_limit=None*, *user_agent=None*, *session=None*, *headers=None*, *timeout=30*)

Bases: `dropbox.dropbox._DropboxTransport`, `dropbox.base_team.DropboxTeamBase`

Use this class to make requests to the Dropbox API using a team's access token. Methods of this class are meant to act on the team, but there is also an *as_user()* method for assuming a team member's identity.

__init__(*oauth2_access_token*, *max_retries_on_error=4*, *max_retries_on_rate_limit=None*, *user_agent=None*, *session=None*, *headers=None*, *timeout=30*)

Parameters

- **oauth2_access_token** (*str*) – OAuth2 access token for making client requests.
- **max_retries_on_error** (*int*) – On 5xx errors, the number of times to retry.

- **max_retries_on_rate_limit** (*Optional[int]*) – On 429 errors, the number of times to retry. If *None*, always retries.
- **user_agent** (*str*) – The user agent to use when making requests. This helps us identify requests coming from your application. We recommend you use the format "AppName/Version". If set, we append "/OfficialDropboxPythonSDKv2/__version__" to the user_agent,
- **session** (requests.sessions.Session) – If not provided, a new session (connection pool) is created. To share a session across multiple clients, use *create_session()*.
- **headers** (*dict*) – Additional headers to add to requests.
- **timeout** (*Optional[float]*) – Maximum duration in seconds that client will wait for any single packet from the server. After the timeout the client will give up on connection. If *None*, client will wait forever. Defaults to 30 seconds.

as_user(*team_member_id*)

Allows a team credential to assume the identity of a member of the team.

Returns A *Dropbox* object that can be used to query on behalf of this member of the team.

Return type *Dropbox*

request(*route*, *namespace*, *request_arg*, *request_binary*, *timeout=None*)

Makes a request to the Dropbox API and in the process validates that the route argument and result are the expected data types. The request_arg is converted to JSON based on the arg_data_type. Likewise, the response is deserialized from JSON and converted to an object based on the {result,error}_data_type.

Parameters

- **host** – The Dropbox API host to connect to.
- **route** (datatypes.stone_base.Route) – The route to make the request to.
- **request_arg** – Argument for the route that conforms to the validator specified by route.arg_type.
- **request_binary** – String or file pointer representing the binary payload. Use None if there is no binary payload.
- **timeout** (*Optional[float]*) – Maximum duration in seconds that client will wait for any single packet from the server. After the timeout the client will give up on connection. If *None*, will use default timeout set on Dropbox object. Defaults to *None*.

Returns The route's result.

request_json_object(*host*, *route_name*, *route_style*, *request_arg*, *request_binary*, *timeout=None*)

Makes a request to the Dropbox API, taking a JSON-serializable Python object as an argument, and returning one as a response.

Parameters

- **host** – The Dropbox API host to connect to.
- **route_name** – The name of the route to invoke.
- **route_style** – The style of the route.
- **request_arg** (*str*) – A JSON-serializable Python object representing the argument for the route.
- **request_binary** (*Optional[bytes]*) – Bytes representing the binary payload. Use None if there is no binary payload.

- **timeout** (*Optional[float]*) – Maximum duration in seconds that client will wait for any single packet from the server. After the timeout the client will give up on connection. If *None*, will use default timeout set on Dropbox object. Defaults to *None*.

Returns The route's result as a JSON-serializable Python object.

request_json_string(*host*, *func_name*, *route_style*, *request_json_arg*, *request_binary*, *timeout=None*)

See `request_json_string_with_retry()` for description of parameters.

request_json_string_with_retry(*host*, *route_name*, *route_style*, *request_json_arg*, *request_binary*, *timeout=None*)

See `request_json_object()` for description of parameters.

Parameters **request_json_arg** – A string representing the serialized JSON argument to the route.

team_alpha_groups_create(*group_name*, *group_external_id=None*, *group_management_type=None*)

Creates a new, empty group, with a requested name. Permission : Team member management

Parameters

- **group_name** (*str*) – Group name.
- **group_external_id** (*Nullable*) – The creator of a team can associate an arbitrary external ID to the group.
- **group_management_type** (*Nullable*) – Whether the team can be managed by selected users, or only by team admins

Return type `dropbox.team.GroupFullInfo`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.team.GroupCreateError`

team_alpha_groups_get_info(*arg*)

Retrieves information about one or more groups. Permission : Team Information

Parameters **arg** (`dropbox.team.GroupsSelector`) – Argument for selecting a list of groups, either by group_ids, or external group IDs.

Return type list

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.team.GroupsGetInfoError`

team_alpha_groups_list(*limit=1000*)

Lists groups on a team. Permission : Team Information

Parameters **limit** (*long*) – Number of results to return per call.

Return type `dropbox.team.GroupsListResult`

team_alpha_groups_list_continue(*cursor*)

Once a cursor has been retrieved from `team_alpha_groups_list()`, use this to paginate through all groups. Permission : Team information

Parameters **cursor** (*str*) – Indicates from what point to get the next set of groups.

Return type `dropbox.team.GroupsListResult`

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.GroupsListContinueError*

team_alpha_groups_update(*group*, *return_members=True*, *new_group_name=None*, *new_group_external_id=None*, *new_group_management_type=None*)

Updates a group's name, external ID or management type. Permission : Team member management

Parameters

- **group** (*dropbox.team.GroupSelector*) – Specify a group.
- **new_group_name** (*Nullable*) – Optional argument. Set group name to this if provided.
- **new_group_external_id** (*Nullable*) – Optional argument. New group external ID. If the argument is None, the group's external_id won't be updated. If the argument is empty string, the group's external id will be cleared.
- **new_group_management_type** (*Nullable*) – Set new group management type, if provided.

Return type *dropbox.team.GroupFullInfo*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.GroupUpdateError*

team_devices_list_member_devices(*team_member_id*, *include_web_sessions=True*, *include_desktop_clients=True*, *include_mobile_clients=True*)

List all device sessions of a team's member.

Parameters

- **team_member_id** (*str*) – The team's member id
- **include_web_sessions** (*bool*) – Whether to list web sessions of the team's member
- **include_desktop_clients** (*bool*) – Whether to list linked desktop devices of the team's member
- **include_mobile_clients** (*bool*) – Whether to list linked mobile devices of the team's member

Return type *dropbox.team.ListMemberDevicesResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.ListMemberDevicesError*

team_devices_list_members_devices(*cursor=None*, *include_web_sessions=True*, *include_desktop_clients=True*, *include_mobile_clients=True*)

List all device sessions of a team.

Parameters

- **cursor** (*Nullable*) – At the first call to the *team_devices_list_members_devices()* the cursor shouldn't be passed. Then, if the result of the call includes a cursor, the following requests should include the received cursors in order to receive the next sub list of team devices

- **include_web_sessions** (*bool*) – Whether to list web sessions of the team members
- **include_desktop_clients** (*bool*) – Whether to list desktop clients of the team members
- **include_mobile_clients** (*bool*) – Whether to list mobile clients of the team members

Return type *dropbox.team.ListMembersDevicesResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.ListMembersDevicesError*

team_devices_list_team_devices(*cursor=None*, *include_web_sessions=True*, *include_desktop_clients=True*, *include_mobile_clients=True*)

List all device sessions of a team.

Parameters

- **cursor** (*Nullable*) – At the first call to the *team_devices_list_team_devices()* the cursor shouldn't be passed. Then, if the result of the call includes a cursor, the following requests should include the received cursors in order to receive the next sub list of team devices
- **include_web_sessions** (*bool*) – Whether to list web sessions of the team members
- **include_desktop_clients** (*bool*) – Whether to list desktop clients of the team members
- **include_mobile_clients** (*bool*) – Whether to list mobile clients of the team members

Return type *dropbox.team.ListTeamDevicesResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.ListTeamDevicesError*

team_devices_revoke_device_session(*arg*)

Revoke a device session of a team's member

Return type None

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.RevokeDeviceSessionError*

team_devices_revoke_device_session_batch(*revoke_devices*)

Revoke a list of device sessions of team members

Return type *dropbox.team.RevokeDeviceSessionBatchResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.RevokeDeviceSessionBatchError*

team_get_info()

Retrieves information about a team.

Return type *dropbox.team.TeamGetInfoResult*

team_groups_create(*group_name*, *group_external_id=None*, *group_management_type=None*)

Creates a new, empty group, with a requested name. Permission : Team member management

Parameters

- **group_name** (*str*) – Group name.
- **group_external_id** (*Nullable*) – The creator of a team can associate an arbitrary external ID to the group.
- **group_management_type** (*Nullable*) – Whether the team can be managed by selected users, or only by team admins

Return type *dropbox.team.GroupFullInfo*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.GroupCreateError*

team_groups_delete(*arg*)

Deletes a group. The group is deleted immediately. However the revoking of group-owned resources may take additional time. Use the *team_groups_job_status_get()* to determine whether this process has completed. Permission : Team member management

Parameters **arg** (*dropbox.team.GroupSelector*) – Argument for selecting a single group, either by group_id or by external group ID.

Return type dropbox.team.LaunchEmptyResult

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.GroupDeleteError*

team_groups_get_info(*arg*)

Retrieves information about one or more groups. Permission : Team Information

Parameters **arg** (*dropbox.team.GroupsSelector*) – Argument for selecting a list of groups, either by group_ids, or external group IDs.

Return type list

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.GroupsGetInfoError*

team_groups_job_status_get(*async_job_id*)

Once an async_job_id is returned from *team_groups_delete()*, *team_groups_members_add()* , or *team_groups_members_remove()* use this method to poll the status of granting/revoking group members' access to group-owned resources. Permission : Team member management

Parameters **async_job_id** (*str*) – Id of the asynchronous job. This is the value of a response returned from the method that launched the job.

Return type dropbox.team.PollEmptyResult

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.GroupsPollError*

team_groups_list(*limit=1000*)
Lists groups on a team. Permission : Team Information

Parameters **limit** (*long*) – Number of results to return per call.

Return type *dropbox.team.GroupsListResult*

team_groups_list_continue(*cursor*)
Once a cursor has been retrieved from *team_groups_list()*, use this to paginate through all groups. Permission : Team information

Parameters **cursor** (*str*) – Indicates from what point to get the next set of groups.

Return type *dropbox.team.GroupsListResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.GroupsListContinueError*

team_groups_members_add(*group*, *members*, *return_members=True*)
Adds members to a group. The members are added immediately. However the granting of group-owned resources may take additional time. Use the *team_groups_job_status_get()* to determine whether this process has completed. Permission : Team member management

Parameters

- **group** (*dropbox.team.GroupSelector*) – Group to which users will be added.
- **members** (*list*) – List of users to be added to the group.

Return type *dropbox.team.GroupMembersChangeResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.GroupMembersAddError*

team_groups_members_list(*group*, *limit=1000*)
Lists members of a group. Permission : Team Information

Parameters

- **group** (*dropbox.team.GroupSelector*) – The group whose members are to be listed.
- **limit** (*long*) – Number of results to return per call.

Return type *dropbox.team.GroupsMembersListResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.GroupSelectorError*

team_groups_members_list_continue(*cursor*)
Once a cursor has been retrieved from *team_groups_members_list()*, use this to paginate through all members of the group. Permission : Team information

Parameters **cursor** (*str*) – Indicates from what point to get the next set of groups.

Return type *dropbox.team.GroupsMembersListResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.GroupsMembersListContinueError*

team_groups_members_remove(*group*, *users*, *return_members=True*)

Removes members from a group. The members are removed immediately. However the revoking of group-owned resources may take additional time. Use the `team_groups_job_status_get()` to determine whether this process has completed. This method permits removing the only owner of a group, even in cases where this is not possible via the web client. Permission : Team member management

Parameters

- **group** (`dropbox.team.GroupSelector`) – Group from which users will be removed.
- **users** (`list`) – List of users to be removed from the group.

Return type `dropbox.team.GroupMembersChangeResult`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.team.GroupMembersRemoveError`

team_groups_members_set_access_type(*group*, *user*, *access_type*, *return_members=True*)

Sets a member's access type in a group. Permission : Team member management

Parameters

- **access_type** (`dropbox.team.GroupAccessType`) – New group access type the user will have.
- **return_members** (`bool`) – Whether to return the list of members in the group. Note that the default value will cause all the group members to be returned in the response. This may take a long time for large groups.

Return type list

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.team.GroupMemberSetAccessTypeError`

team_groups_update(*group*, *return_members=True*, *new_group_name=None*, *new_group_external_id=None*, *new_group_management_type=None*)

Updates a group's name and/or external ID. Permission : Team member management

Parameters

- **group** (`dropbox.team.GroupSelector`) – Specify a group.
- **new_group_name** (`Nullable`) – Optional argument. Set group name to this if provided.
- **new_group_external_id** (`Nullable`) – Optional argument. New group external ID. If the argument is None, the group's external_id won't be updated. If the argument is empty string, the group's external id will be cleared.
- **new_group_management_type** (`Nullable`) – Set new group management type, if provided.

Return type `dropbox.team.GroupFullInfo`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.team.GroupUpdateError`

team_linked_apps_list_member_linked_apps(*team_member_id*)

List all linked applications of the team member. Note, this endpoint does not list any team-linked applications.

Parameters **team_member_id** (*str*) – The team member id

Return type *dropbox.team.ListMemberAppsResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.ListMemberAppsError*

team_linked_apps_list_members_linked_apps(*cursor=None*)

List all applications linked to the team members' accounts. Note, this endpoint does not list any team-linked applications.

Parameters **cursor** (*Nullable*) – At the first call to the *team_linked_apps_list_members_linked_apps()* the cursor shouldn't be passed. Then, if the result of the call includes a cursor, the following requests should include the received cursors in order to receive the next sub list of the team applications

Return type *dropbox.team.ListMembersAppsResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.ListMembersAppsError*

team_linked_apps_list_team_linked_apps(*cursor=None*)

List all applications linked to the team members' accounts. Note, this endpoint doesn't list any team-linked applications.

Parameters **cursor** (*Nullable*) – At the first call to the *team_linked_apps_list_team_linked_apps()* the cursor shouldn't be passed. Then, if the result of the call includes a cursor, the following requests should include the received cursors in order to receive the next sub list of the team applications

Return type *dropbox.team.ListTeamAppsResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.ListTeamAppsError*

team_linked_apps_revoke_linked_app(*app_id*, *team_member_id*, *keep_app_folder=True*)

Revoke a linked application of the team member

Parameters

- **app_id** (*str*) – The application's unique id
- **team_member_id** (*str*) – The unique id of the member owning the device
- **keep_app_folder** (*bool*) – Whether to keep the application dedicated folder (in case the application uses one)

Return type None

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.RevokeLinkedAppError*

team_linked_apps_revoke_linked_app_batch(*revoke_linked_app*)
Revoke a list of linked applications of the team members

Return type *dropbox.team.RevokeLinkedAppBatchResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.RevokeLinkedAppBatchError*

team_members_add(*new_members*, *force_async=False*)
Adds members to a team. Permission : Team member management A maximum of 20 members can be specified in a single call. If no Dropbox account exists with the email address specified, a new Dropbox account will be created with the given email address, and that account will be invited to the team. If a personal Dropbox account exists with the email address specified in the call, this call will create a placeholder Dropbox account for the user on the team and send an email inviting the user to migrate their existing personal account onto the team. Team member management apps are required to set an initial given_name and surname for a user to use in the team invitation and for 'Perform as team member' actions taken on the user before they become 'active'.

Parameters

- **new_members** (*list*) – Details of new members to be added to the team.
- **force_async** (*bool*) – Whether to force the add to happen asynchronously.

Return type *dropbox.team.MembersAddLaunch*

team_members_add_job_status_get(*async_job_id*)
Once an async_job_id is returned from *team_members_add()* , use this to poll the status of the asynchronous request. Permission : Team member management

Parameters **async_job_id** (*str*) – Id of the asynchronous job. This is the value of a response returned from the method that launched the job.

Return type *dropbox.team.MembersAddJobStatus*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: `dropbox.team.PollError`

team_members_get_info(*members*)
Returns information about multiple team members. Permission : Team information This endpoint will return `MembersGetInfoItem.id_not_found`, for IDs (or emails) that cannot be matched to a valid team member.

Parameters **members** (*list*) – List of team members.

Return type list

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.MembersGetInfoError*

team_members_list(*limit=1000*, *include_removed=False*)
Lists members of a team. Permission : Team information

Parameters

- **limit** (*long*) – Number of results to return per call.
- **include_removed** (*bool*) – Whether to return removed members.

Return type *dropbox.team.MembersListResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.MembersListError*

team_members_list_continue(*cursor*)

Once a cursor has been retrieved from *team_members_list()*, use this to paginate through all team members. Permission : Team information

Parameters **cursor** (*str*) – Indicates from what point to get the next set of members.

Return type *dropbox.team.MembersListResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.MembersListContinueError*

team_members_recover(*user*)

Recover a deleted member. Permission : Team member management Exactly one of team_member_id, email, or external_id must be provided to identify the user account.

Parameters **user** (*dropbox.team.UserSelectorArg*) – Identity of user to recover.

Return type None

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.MembersRecoverError*

team_members_remove(*user*, *wipe_data=True*, *transfer_dest_id=None*, *transfer_admin_id=None*, *keep_account=False*)

Removes a member from a team. Permission : Team member management Exactly one of team_member_id, email, or external_id must be provided to identify the user account. This is not a deactivation where the account can be re-activated again. Calling *team_members_add()* with the removed user's email address will create a new account with a new team_member_id that will not have access to any content that was shared with the initial account. This endpoint may initiate an asynchronous job. To obtain the final result of the job, the client should periodically poll *team_members_remove_job_status_get()*.

Parameters

- **transfer_dest_id** (*Nullable*) – If provided, files from the deleted member account will be transferred to this user.
- **transfer_admin_id** (*Nullable*) – If provided, errors during the transfer process will be sent via email to this user. If the transfer_dest_id argument was provided, then this argument must be provided as well.
- **keep_account** (*bool*) – Downgrade the member to a Basic account. The user will retain the email address associated with their Dropbox account and data in their account that is not restricted to team members.

Return type dropbox.team.LaunchEmptyResult

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.MembersRemoveError*

team_members_remove_job_status_get(*async_job_id*)

Once an async_job_id is returned from `team_members_remove()` , use this to poll the status of the asynchronous request. Permission : Team member management

Parameters **async_job_id** (*str*) – Id of the asynchronous job. This is the value of a response returned from the method that launched the job.

Return type `dropbox.team.PollEmptyResult`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.team.PollError`

team_members_send_welcome_email(*arg*)

Sends welcome email to pending team member. Permission : Team member management Exactly one of team_member_id, email, or external_id must be provided to identify the user account. No-op if team member is not pending.

Parameters **arg** (`dropbox.team.UserSelectorArg`) – Argument for selecting a single user, either by team_member_id, external_id or email.

Return type None

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.team.MembersSendWelcomeError`

team_members_set_admin_permissions(*user*, *new_role*)

Updates a team member's permissions. Permission : Team member management

Parameters

- **user** (`dropbox.team.UserSelectorArg`) – Identity of user whose role will be set.
- **new_role** (`dropbox.team.AdminTier`) – The new role of the member.

Return type `dropbox.team.MembersSetPermissionsResult`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: `dropbox.team.MembersSetPermissionsError`

team_members_set_profile(*user*, *new_email=None*, *new_external_id=None*, *new_given_name=None*, *new_surname=None*)

Updates a team member's profile. Permission : Team member management

Parameters

- **user** (`dropbox.team.UserSelectorArg`) – Identity of user whose profile will be set.
- **new_email** (`Nullable`) – New email for member.
- **new_external_id** (`Nullable`) – New external ID for member.
- **new_given_name** (`Nullable`) – New given name for member.
- **new_surname** (`Nullable`) – New surname for member.

Return type `dropbox.team.TeamMemberInfo`

Raises `dropbox.exceptions.ApiError`

If this raises, ApiError.reason is of type: *dropbox.team.MembersSetProfileError*

team_members_suspend(*user*, *wipe_data=True*)

Suspend a member from a team. Permission : Team member management Exactly one of team_member_id, email, or external_id must be provided to identify the user account.

Parameters

- **user** (*dropbox.team.UserSelectorArg*) – Identity of user to remove/suspend.
- **wipe_data** (*bool*) – If provided, controls if the user's data will be deleted on their linked devices.

Return type None

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.MembersSuspendError*

team_members_unsuspend(*user*)

Unsuspend a member from a team. Permission : Team member management Exactly one of team_member_id, email, or external_id must be provided to identify the user account.

Parameters **user** (*dropbox.team.UserSelectorArg*) – Identity of user to unsuspend.

Return type None

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.MembersUnsuspendError*

team_properties_template_add(*name*, *description*, *fields*)

Add a property template. See route files/properties/add to add properties to a file.

Return type *dropbox.team.AddPropertyTemplateResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: `dropbox.team.ModifyPropertyTemplateError`

team_properties_template_get(*template_id*)

Get the schema for a specified template.

Parameters **template_id** (*str*) – An identifier for property template added by route properties/template/add.

Return type `dropbox.team.GetPropertyTemplateResult`

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: `dropbox.team.PropertyTemplateError`

team_properties_template_list()

Get the property template identifiers for a team. To get the schema of each template use *team_properties_template_get()*.

Return type `dropbox.team.ListPropertyTemplateIds`

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: `dropbox.team.PropertyTemplateError`

team_properties_template_update(*template_id*, *name=None*, *description=None*, *add_fields=None*)

Update a property template. This route can update the template name, the template description and add optional properties to templates.

Parameters

- **template_id** (*str*) – An identifier for property template added by *team_properties_template_add()*.
- **name** (*Nullable*) – A display name for the property template. Property template names can be up to 256 bytes.
- **description** (*Nullable*) – Description for new property template. Property template descriptions can be up to 1024 bytes.
- **add_fields** (*Nullable*) – This is a list of custom properties to add to the property template. There can be up to 64 properties in a single property template.

Return type *dropbox.team.UpdatePropertyTemplateResult*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: `dropbox.team.ModifyPropertyTemplateError`

team_reports_get_activity(*start_date=None*, *end_date=None*)

Retrieves reporting data about a team's user activity.

Parameters

- **start_date** (*Nullable*) – Optional starting date (inclusive)
- **end_date** (*Nullable*) – Optional ending date (exclusive)

Return type *dropbox.team.GetActivityReport*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.DateRangeError*

team_reports_get_devices(*start_date=None*, *end_date=None*)

Retrieves reporting data about a team's linked devices.

Parameters

- **start_date** (*Nullable*) – Optional starting date (inclusive)
- **end_date** (*Nullable*) – Optional ending date (exclusive)

Return type *dropbox.team.GetDevicesReport*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.DateRangeError*

team_reports_get_membership(*start_date=None*, *end_date=None*)

Retrieves reporting data about a team's membership.

Parameters

- **start_date** (*Nullable*) – Optional starting date (inclusive)
- **end_date** (*Nullable*) – Optional ending date (exclusive)

Return type *dropbox.team.GetMembershipReport*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.DateRangeError*

team_reports_get_storage(*start_date=None*, *end_date=None*)
Retrieves reporting data about a team's storage usage.

Parameters

- **start_date** (*Nullable*) – Optional starting date (inclusive)
- **end_date** (*Nullable*) – Optional ending date (exclusive)

Return type *dropbox.team.GetStorageReport*

Raises *dropbox.exceptions.ApiError*

If this raises, ApiError.reason is of type: *dropbox.team.DateRangeError*

dropbox.dropbox.**create_session**(*max_connections=8*, *proxies=None*)
Creates a session object that can be used by multiple *Dropbox* and *DropboxTeam* instances. This lets you share a connection pool amongst them, as well as proxy parameters.

Parameters

- **max_connections** (*int*) – Maximum connection pool size.
- **proxies** (*dict*) – See the requests module for more details.

Return type `requests.sessions.Session`. See the requests module for more details.

2.3 dropbox.async – Async

class dropbox.async.**LaunchEmptyResult**(*tag*, *value=None*)
Bases: *dropbox.async.LaunchResultBase*

Result returned by methods that may either launch an asynchronous job or complete synchronously. Upon synchronous completion of the job, no additional information is returned.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***complete*** – The job finished synchronously and successfully.

complete = LaunchEmptyResult('complete', None)

is_complete()
Check if the union tag is `complete`.

Return type bool

class dropbox.async.**LaunchResultBase**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Result returned by methods that launch an asynchronous job. A method who may either launch an asynchronous job, or complete the request synchronously, can use this union by extending it, and adding a 'complete' field with the type of the synchronous response. See *LaunchEmptyResult* for an example.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***async_job_id*** (*str*) – This response indicates that the processing is asynchronous. The string is an id that can be used to obtain the status of the asynchronous job.

classmethod **`async_job_id`**(*val*)

Create an instance of this class set to the `async_job_id` tag with value `val`.

Parameters **`val`** (*str*) –

Return type *LaunchResultBase*

`get_async_job_id`()

This response indicates that the processing is asynchronous. The string is an id that can be used to obtain the status of the asynchronous job.

Only call this if *`is_async_job_id()`* is true.

Return type str

`is_async_job_id`()

Check if the union tag is `async_job_id`.

Return type bool

class `dropbox.async.`**`PollArg`**(*async_job_id=None*)

Bases: `object`

Arguments for methods that poll the status of an asynchronous job.

Variables ***async_job_id*** – Id of the asynchronous job. This is the value of a response returned from the method that launched the job.

`__init__`(*async_job_id=None*)

`async_job_id`

Id of the asynchronous job. This is the value of a response returned from the method that launched the job.

Return type str

class `dropbox.async.`**`PollEmptyResult`**(*tag*, *value=None*)

Bases: *`dropbox.async.PollResultBase`*

Result returned by methods that poll for the status of an asynchronous job. Upon completion of the job, no additional information is returned.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***complete*** – The asynchronous job has completed successfully.

`complete` = PollEmptyResult('complete', None)

`is_complete`()

Check if the union tag is `complete`.

Return type bool

class `dropbox.async.`**`PollError`**(*tag*, *value=None*)

Bases: `dropbox.stone_base.Union`

Error returned by methods for polling the status of asynchronous job.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***invalid_async_job_id*** – The job ID is invalid.

- ***internal_error*** – Something went wrong with the job on Dropbox's end. You'll need to verify that the action you were taking succeeded, and if not, try again. This should happen very rarely.

internal_error = PollError('internal_error', None)

invalid_async_job_id = PollError('invalid_async_job_id', None)

is_internal_error()
Check if the union tag is `internal_error`.

Return type bool

is_invalid_async_job_id()
Check if the union tag is `invalid_async_job_id`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

other = PollError('other', None)

class dropbox.async.**PollResultBase**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Result returned by methods that poll for the status of an asynchronous job. Unions that extend this union should add a 'complete' field with a type of the information returned upon job completion. See `PollEmptyResult` for an example.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***in_progress*** – The asynchronous job is still in progress.

in_progress = PollResultBase('in_progress', None)

is_in_progress()
Check if the union tag is `in_progress`.

Return type bool

2.4 dropbox.files – Files

This namespace contains endpoints and data types for basic file operations.

class dropbox.files.**AddPropertiesError**(*tag*, *value=None*)
Bases: `dropbox.files.InvalidPropertyGroupError`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***property_group_already_exists*** – This property group already exists for this file.

is_property_group_already_exists()
Check if the union tag is `property_group_already_exists`.

Return type bool

property_group_already_exists = AddPropertiesError('property_group_already_exists', None)

class dropbox.files.**AlphaGetMetadataArg**(*path=None*, *include_media_info=None*, *include_deleted=None*, *include_has_explicit_shared_members=None*, *include_property_templates=None*)

Bases: *dropbox.files.GetMetadataArg*

Variables ***include_property_templates*** – If set to a valid list of template IDs, `FileMetadata.property_groups` is set for files with custom properties.

include_property_templates

If set to a valid list of template IDs, `FileMetadata.property_groups` is set for files with custom properties.

Return type list of [str]

class dropbox.files.**AlphaGetMetadataError**(*tag*, *value=None*)

Bases: *dropbox.files.GetMetadataError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

get_properties_error()

Only call this if *is_properties_error()* is true.

Return type *LookUpPropertiesError*

is_properties_error()

Check if the union tag is `properties_error`.

Return type bool

classmethod properties_error(*val*)

Create an instance of this class set to the `properties_error` tag with value `val`.

Parameters **val** (`LookUpPropertiesError`) –

Return type *AlphaGetMetadataError*

class dropbox.files.**CommitInfo**(*path=None*, *mode=None*, *autorename=None*, *client_modified=None*, *mute=None*)

Bases: `object`

Variables

- ***path*** – Path in the user's Dropbox to save the file.
- ***mode*** – Selects what to do if the file already exists.
- ***autorename*** – If there's a conflict, as determined by `mode`, have the Dropbox server try to autorename the file to avoid conflict.
- ***client_modified*** – The value to store as the `client_modified` timestamp. Dropbox automatically records the time at which the file was written to the Dropbox servers. It can also record an additional timestamp, provided by Dropbox desktop clients, mobile clients, and API apps of when the file was actually created or modified.
- ***mute*** – Normally, users are made aware of any file modifications in their Dropbox account via notifications in the client software. If `True`, this tells the clients that this modification shouldn't result in a user notification.

autorename

If there's a conflict, as determined by `mode`, have the Dropbox server try to autorename the file to avoid conflict.

Return type bool

client_modified
The value to store as the `client_modified` timestamp. Dropbox automatically records the time at which the file was written to the Dropbox servers. It can also record an additional timestamp, provided by Dropbox desktop clients, mobile clients, and API apps of when the file was actually created or modified.

Return type datetime.datetime

mode
Selects what to do if the file already exists.

Return type *WriteMode*

mute
Normally, users are made aware of any file modifications in their Dropbox account via notifications in the client software. If `True`, this tells the clients that this modification shouldn't result in a user notification.

Return type bool

path
Path in the user's Dropbox to save the file.

Return type str

class `dropbox.files.`**`CommitInfoWithProperties`**(*path=None*, *mode=None*, *autorename=None*, *client_modified=None*, *mute=None*, *property_groups=None*)
Bases: *`dropbox.files.CommitInfo`*

Variables ***`property_groups`*** – List of custom properties to add to file.

property_groups
List of custom properties to add to file.

Return type list of [properties.PropertyGroup_validator]

class `dropbox.files.`**`CreateFolderArg`**(*path=None*, *autorename=None*)
Bases: `object`

Variables

- ***`path`*** – Path in the user's Dropbox to create.
- ***`autorename`*** – If there's a conflict, have the Dropbox server try to autorename the folder to avoid the conflict.

autorename
If there's a conflict, have the Dropbox server try to autorename the folder to avoid the conflict.

Return type bool

path
Path in the user's Dropbox to create.

Return type str

class `dropbox.files.`**`CreateFolderError`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

get_path()
Only call this if *`is_path()`* is true.

Return type *WriteError*

is_path()
Check if the union tag is `path`.

Return type bool

classmethod path(*val*)
Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (`WriteError`) –

Return type *CreateFolderError*

class dropbox.files.**DeleteArg**(*path=None*)
Bases: `object`

Variables ***path*** – Path in the user's Dropbox to delete.

path
Path in the user's Dropbox to delete.

Return type str

class dropbox.files.**DeleteBatchArg**(*entries=None*)
Bases: `object`

entries

Return type list of [DeleteArg]

class dropbox.files.**DeleteBatchError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***too_many_write_operations*** – There are too many write operations in user's Dropbox. Please retry this request.

is_other()
Check if the union tag is `other`.

Return type bool

is_too_many_write_operations()
Check if the union tag is `too_many_write_operations`.

Return type bool

other = DeleteBatchError('other', None)

too_many_write_operations = DeleteBatchError('too_many_write_operations', None)

class dropbox.files.**DeleteBatchJobStatus**(*tag*, *value=None*)
Bases: *dropbox.async.PollResultBase*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***complete*** (`DeleteBatchResult`) – The batch delete has finished.
- ***failed*** (`DeleteBatchError`) – The batch delete has failed.

classmethod complete(*val*)
Create an instance of this class set to the `complete` tag with value `val`.

Parameters **val** (DeleteBatchResult) –

Return type *DeleteBatchJobStatus*

classmethod failed(*val*)
Create an instance of this class set to the `failed` tag with value `val`.

Parameters **val** (DeleteBatchError) –

Return type *DeleteBatchJobStatus*

get_complete()
The batch delete has finished.

Only call this if *is_complete()* is true.

Return type *DeleteBatchResult*

get_failed()
The batch delete has failed.

Only call this if *is_failed()* is true.

Return type *DeleteBatchError*

is_complete()
Check if the union tag is `complete`.

Return type bool

is_failed()
Check if the union tag is `failed`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

other = DeleteBatchJobStatus('other', None)

class dropbox.files.**DeleteBatchResult**(*entries=None*)
Bases: `object`

entries

Return type list of [DeleteBatchResultEntry]

class dropbox.files.**DeleteBatchResultEntry**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

classmethod failure(*val*)
Create an instance of this class set to the `failure` tag with value `val`.

Parameters **val** (DeleteError) –

Return type *DeleteBatchResultEntry*

get_failure()
Only call this if *is_failure()* is true.

Return type *DeleteError*

get_success()
Only call this if *is_success()* is true.

Return type *DeleteResult*

is_failure()
Check if the union tag is `failure`.

Return type bool

is_success()
Check if the union tag is `success`.

Return type bool

classmethod success(*val*)
Create an instance of this class set to the `success` tag with value `val`.

Parameters **val** (`DeleteResult`) –

Return type *DeleteBatchResultEntry*

class `dropbox.files.`**DeleteError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

get_path_lookup()
Only call this if *is_path_lookup()* is true.

Return type *LookupError*

get_path_write()
Only call this if *is_path_write()* is true.

Return type *WriteError*

is_other()
Check if the union tag is `other`.

Return type bool

is_path_lookup()
Check if the union tag is `path_lookup`.

Return type bool

is_path_write()
Check if the union tag is `path_write`.

Return type bool

other = DeleteError('other', None)

classmethod path_lookup(*val*)
Create an instance of this class set to the `path_lookup` tag with value `val`.

Parameters **val** (`LookupError`) –

Return type *DeleteError*

classmethod path_write(*val*)
Create an instance of this class set to the `path_write` tag with value `val`.

Parameters **val** (`WriteError`) –

Return type *DeleteError*

class `dropbox.files.`**`DeleteResult`**(*metadata=None*)
Bases: `object`

`metadata`

Return type *Metadata*

class `dropbox.files.`**`DeletedMetadata`**(*name=None*, *path_lower=None*, *path_display=None*, *parent_shared_folder_id=None*)
Bases: *`dropbox.files.Metadata`*

Indicates that there used to be a file or folder at this path, but it no longer exists.

class `dropbox.files.`**`Dimensions`**(*height=None*, *width=None*)
Bases: `object`

Dimensions for a photo or video.

Variables

- ***`height`*** – Height of the photo/video.
- ***`width`*** – Width of the photo/video.

`height`
Height of the photo/video.

Return type long

`width`
Width of the photo/video.

Return type long

class `dropbox.files.`**`DownloadArg`**(*path=None*, *rev=None*)
Bases: `object`

Variables

- ***`path`*** – The path of the file to download.
- ***`rev`*** – Deprecated. Please specify revision in `path` instead.

`path`
The path of the file to download.

Return type str

`rev`
Deprecated. Please specify revision in `path` instead.

Return type str

class `dropbox.files.`**`DownloadError`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

`get_path`()
Only call this if *`is_path()`* is true.

Return type *LookupError*

`is_other`()
Check if the union tag is `other`.

Return type bool

is_path()

Check if the union tag is `path`.

Return type bool

other = DownloadError('other', None)

classmethod path(*val*)

Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (`LookupError`) –

Return type *DownloadError*

class dropbox.files.**FileMetadata**(*name=None, id=None, client_modified=None, server_modified=None, rev=None, size=None, path_lower=None, path_display=None, parent_shared_folder_id=None, media_info=None, sharing_info=None, property_groups=None, has_explicit_shared_members=None*)

Bases: *dropbox.files.Metadata*

Variables

- ***id*** – A unique identifier for the file.
- ***client_modified*** – For files, this is the modification time set by the desktop client when the file was added to Dropbox. Since this time is not verified (the Dropbox server stores whatever the desktop client sends up), this should only be used for display purposes (such as sorting) and not, for example, to determine if a file has changed or not.
- ***server_modified*** – The last time the file was modified on Dropbox.
- ***rev*** – A unique identifier for the current revision of a file. This field is the same rev as elsewhere in the API and can be used to detect changes and avoid conflicts.
- ***size*** – The file size in bytes.
- ***media_info*** – Additional information if the file is a photo or video.
- ***sharing_info*** – Set if this file is contained in a shared folder.
- ***property_groups*** – Additional information if the file has custom properties with the property template specified.
- ***has_explicit_shared_members*** – This flag will only be present if include_has_explicit_shared_members is true in *dropbox.dropbox.Dropbox.files_list_folder()* or *dropbox.dropbox.Dropbox.files_get_metadata()*. If this flag is present, it will be true if this file has any explicit shared members. This is different from sharing_info in that this could be true in the case where a file has explicit members but is not contained within a shared folder.

client_modified

For files, this is the modification time set by the desktop client when the file was added to Dropbox. Since this time is not verified (the Dropbox server stores whatever the desktop client sends up), this should only be used for display purposes (such as sorting) and not, for example, to determine if a file has changed or not.

Return type datetime.datetime

has_explicit_shared_members

This flag will only be present if include_has_explicit_shared_members is true in `dropbox.dropbox.Dropbox.files_list_folder()` or `dropbox.dropbox.Dropbox.files_get_metadata()`. If this flag is present, it will be true if this file has any explicit shared members. This is different from sharing_info in that this could be true in the case where a file has explicit members but is not contained within a shared folder.

Return type bool

id

A unique identifier for the file.

Return type str

media_info

Additional information if the file is a photo or video.

Return type *MediaInfo*

property_groups

Additional information if the file has custom properties with the property template specified.

Return type list of [properties.PropertyGroup_validator]

rev

A unique identifier for the current revision of a file. This field is the same rev as elsewhere in the API and can be used to detect changes and avoid conflicts.

Return type str

server_modified

The last time the file was modified on Dropbox.

Return type datetime.datetime

sharing_info

Set if this file is contained in a shared folder.

Return type *FileSharingInfo*

size

The file size in bytes.

Return type long

class `dropbox.files.`**`FileSharingInfo`**(*read_only=None*, *parent_shared_folder_id=None*, *modified_by=None*)

Bases: `dropbox.files.SharingInfo`

Sharing info for a file which is contained by a shared folder.

Variables

- ***parent_shared_folder_id*** – ID of shared folder that holds this file.
- ***modified_by*** – The last user who modified the file. This field will be null if the user's account has been deleted.

modified_by

The last user who modified the file. This field will be null if the user's account has been deleted.

Return type str

parent_shared_folder_id

ID of shared folder that holds this file.

Return type str

class `dropbox.files.`**`FolderMetadata`**(*name=None*, *id=None*, *path_lower=None*, *path_display=None*, *parent_shared_folder_id=None*, *shared_folder_id=None*, *sharing_info=None*, *property_groups=None*)

Bases: `dropbox.files.Metadata`

Variables

- ***id*** – A unique identifier for the folder.
- ***shared_folder_id*** – Deprecated. Please use `sharing_info` instead.
- ***sharing_info*** – Set if the folder is contained in a shared folder or is a shared folder mount point.
- ***property_groups*** – Additional information if the file has custom properties with the property template specified.

`id`

A unique identifier for the folder.

Return type str

`property_groups`

Additional information if the file has custom properties with the property template specified.

Return type list of [properties.PropertyGroup_validator]

`shared_folder_id`

Deprecated. Please use `sharing_info` instead.

Return type str

`sharing_info`

Set if the folder is contained in a shared folder or is a shared folder mount point.

Return type *FolderSharingInfo*

class `dropbox.files.`**`FolderSharingInfo`**(*read_only=None*, *parent_shared_folder_id=None*, *shared_folder_id=None*, *traverse_only=None*, *no_access=None*)

Bases: `dropbox.files.SharingInfo`

Sharing info for a folder which is contained in a shared folder or is a shared folder mount point.

Variables

- ***parent_shared_folder_id*** – Set if the folder is contained by a shared folder.
- ***shared_folder_id*** – If this folder is a shared folder mount point, the ID of the shared folder mounted at this location.
- ***traverse_only*** – Specifies that the folder can only be traversed and the user can only see a limited subset of the contents of this folder because they don't have read access to this folder. They do, however, have access to some sub folder.
- ***no_access*** – Specifies that the folder cannot be accessed by the user.

`no_access`

Specifies that the folder cannot be accessed by the user.

Return type bool

parent_shared_folder_id
Set if the folder is contained by a shared folder.

Return type str

shared_folder_id
If this folder is a shared folder mount point, the ID of the shared folder mounted at this location.

Return type str

traverse_only
Specifies that the folder can only be traversed and the user can only see a limited subset of the contents of this folder because they don't have read access to this folder. They do, however, have access to some sub folder.

Return type bool

class dropbox.files.**GetCopyReferenceArg**(*path=None*)
Bases: object

Variables ***path*** – The path to the file or folder you want to get a copy reference to.

path
The path to the file or folder you want to get a copy reference to.

Return type str

class dropbox.files.**GetCopyReferenceError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

get_path()
Only call this if *is_path()* is true.

Return type *LookupError*

is_other()
Check if the union tag is other.

Return type bool

is_path()
Check if the union tag is path.

Return type bool

other = GetCopyReferenceError('other', None)

classmethod path(*val*)
Create an instance of this class set to the path tag with value val.

Parameters **val** (LookupError) –

Return type *GetCopyReferenceError*

class dropbox.files.**GetCopyReferenceResult**(*metadata=None*, *copy_reference=None*, *expires=None*)
Bases: object

Variables

- ***metadata*** – Metadata of the file or folder.
- ***copy_reference*** – A copy reference to the file or folder.

- ***expires*** – The expiration date of the copy reference. This value is currently set to be far enough in the future so that expiration is effectively not an issue.

copy_reference
A copy reference to the file or folder.

Return type str

expires
The expiration date of the copy reference. This value is currently set to be far enough in the future so that expiration is effectively not an issue.

Return type datetime.datetime

metadata
Metadata of the file or folder.

Return type *Metadata*

class `dropbox.files.`**`GetMetadataArg`**(*path=None*, *include_media_info=None*, *include_deleted=None*, *include_has_explicit_shared_members=None*)

Bases: `object`

Variables

- ***path*** – The path of a file or folder on Dropbox.
- ***include_media_info*** – If true, `FileMetadata.media_info` is set for photo and video.
- ***include_deleted*** – If true, *`DeletedMetadata`* will be returned for deleted file or folder, otherwise `LookupError.not_found` will be returned.
- ***include_has_explicit_shared_members*** – If true, the results will include a flag for each file indicating whether or not that file has any explicit members.

include_deleted
If true, *`DeletedMetadata`* will be returned for deleted file or folder, otherwise `LookupError.not_found` will be returned.

Return type bool

include_has_explicit_shared_members
If true, the results will include a flag for each file indicating whether or not that file has any explicit members.

Return type bool

include_media_info
If true, `FileMetadata.media_info` is set for photo and video.

Return type bool

path
The path of a file or folder on Dropbox.

Return type str

class `dropbox.files.`**`GetMetadataError`**(*tag*, *value=None*)

Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

get_path()
Only call this if *`is_path()`* is true.

Return type *LookupError*

is_path()
Check if the union tag is `path`.

Return type bool

classmethod path(*val*)
Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (`LookupError`) –

Return type *GetMetadataError*

class dropbox.files.**GetTemporaryLinkArg**(*path=None*)
Bases: `object`

Variables ***path*** – The path to the file you want a temporary link to.

path
The path to the file you want a temporary link to.

Return type str

class dropbox.files.**GetTemporaryLinkError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

get_path()
Only call this if *`is_path()`* is true.

Return type *LookupError*

is_other()
Check if the union tag is `other`.

Return type bool

is_path()
Check if the union tag is `path`.

Return type bool

other = GetTemporaryLinkError('other', None)

classmethod path(*val*)
Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (`LookupError`) –

Return type *GetTemporaryLinkError*

class dropbox.files.**GetTemporaryLinkResult**(*metadata=None*, *link=None*)
Bases: `object`

Variables

- ***metadata*** – Metadata of the file.
- ***link*** – The temporary link which can be used to stream content the file.

link
The temporary link which can be used to stream content the file.

Return type str

metadata
Metadata of the file.

Return type *FileMetadata*

class dropbox.files.**GpsCoordinates**(*latitude=None*, *longitude=None*)
Bases: object

GPS coordinates for a photo or video.

Variables

- ***latitude*** – Latitude of the GPS coordinates.
- ***longitude*** – Longitude of the GPS coordinates.

latitude
Latitude of the GPS coordinates.

Return type float

longitude
Longitude of the GPS coordinates.

Return type float

class dropbox.files.**InvalidPropertyGroupError**(*tag*, *value=None*)
Bases: *dropbox.files.PropertiesError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***property_field_too_large*** – A field value in this property group is too large.
- ***does_not_fit_template*** – The property group specified does not conform to the property template.

does_not_fit_template = InvalidPropertyGroupError('does_not_fit_template', None)

is_does_not_fit_template()
Check if the union tag is `does_not_fit_template`.

Return type bool

is_property_field_too_large()
Check if the union tag is `property_field_too_large`.

Return type bool

property_field_too_large = InvalidPropertyGroupError('property_field_too_large', None)

class dropbox.files.**ListFolderArg**(*path=None*, *recursive=None*, *include_media_info=None*, *include_deleted=None*, *include_has_explicit_shared_members=None*)
Bases: object

Variables

- ***path*** – The path to the folder you want to see the contents of.
- ***recursive*** – If true, the list folder operation will be applied recursively to all subfolders and the response will contain contents of all subfolders.
- ***include_media_info*** – If true, `FileMetadata.media_info` is set for photo and video.

- *include_deleted* – If true, the results will include entries for files and folders that used to exist but were deleted.
- *include_has_explicit_shared_members* – If true, the results will include a flag for each file indicating whether or not that file has any explicit members.

include_deleted

If true, the results will include entries for files and folders that used to exist but were deleted.

Return type bool

include_has_explicit_shared_members

If true, the results will include a flag for each file indicating whether or not that file has any explicit members.

Return type bool

include_media_info

If true, `FileMetadata.media_info` is set for photo and video.

Return type bool

path

The path to the folder you want to see the contents of.

Return type str

recursive

If true, the list folder operation will be applied recursively to all subfolders and the response will contain contents of all subfolders.

Return type bool

class `dropbox.files.`**`ListFolderContinueArg`**(*cursor=None*)

Bases: `object`

Variables *cursor* – The cursor returned by your last call to *dropbox.dropbox.Dropbox.files_list_folder()* or *dropbox.dropbox.Dropbox.files_list_folder_continue()*.

cursor

The cursor returned by your last call to *dropbox.dropbox.Dropbox.files_list_folder()* or *dropbox.dropbox.Dropbox.files_list_folder_continue()*.

Return type str

class `dropbox.files.`**`ListFolderContinueError`**(*tag*, *value=None*)

Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables *reset* – Indicates that the cursor has been invalidated. Call *dropbox.dropbox.Dropbox.files_list_folder()* to obtain a new cursor.

get_path()

Only call this if *is_path()* is true.

Return type *LookupError*

is_other()

Check if the union tag is `other`.

Return type bool

is_path()
Check if the union tag is `path`.

Return type bool

is_reset()
Check if the union tag is `reset`.

Return type bool

other = ListFolderContinueError('other', None)

classmethod path(*val*)
Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (`LookupError`) –

Return type *ListFolderContinueError*

reset = ListFolderContinueError('reset', None)

class dropbox.files.**ListFolderError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

get_path()
Only call this if *is_path()* is true.

Return type *LookupError*

is_other()
Check if the union tag is `other`.

Return type bool

is_path()
Check if the union tag is `path`.

Return type bool

other = ListFolderError('other', None)

classmethod path(*val*)
Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (`LookupError`) –

Return type *ListFolderError*

class dropbox.files.**ListFolderGetLatestCursorResult**(*cursor=None*)
Bases: `object`

Variables ***cursor*** – Pass the cursor into *dropbox.dropbox.Dropbox.files_list_folder_continue()* to see what's changed in the folder since your previous query.

cursor
Pass the cursor into *dropbox.dropbox.Dropbox.files_list_folder_continue()* to see what's changed in the folder since your previous query.

Return type str

class dropbox.files.**ListFolderLongpollArg**(*cursor=None*, *timeout=None*)
Bases: `object`

Variables

- ***cursor*** – A cursor as returned by *dropbox.dropbox.Dropbox.files_list_folder()* or *dropbox.dropbox.Dropbox.files_list_folder_continue()*. Cursors retrieved by setting `ListFolderArg.include_media_info` to `True` are not supported.
- ***timeout*** – A timeout in seconds. The request will block for at most this length of time, plus up to 90 seconds of random jitter added to avoid the thundering herd problem. Care should be taken when using this parameter, as some network infrastructure does not support long timeouts.

cursor

A cursor as returned by *dropbox.dropbox.Dropbox.files_list_folder()* or *dropbox.dropbox.Dropbox.files_list_folder_continue()*. Cursors retrieved by setting `ListFolderArg.include_media_info` to `True` are not supported.

Return type str

timeout

A timeout in seconds. The request will block for at most this length of time, plus up to 90 seconds of random jitter added to avoid the thundering herd problem. Care should be taken when using this parameter, as some network infrastructure does not support long timeouts.

Return type long

class `dropbox.files.`**`ListFolderLongpollError`**(*tag*, *value=None*)

Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***reset*** – Indicates that the cursor has been invalidated. Call *dropbox.dropbox.Dropbox.files_list_folder()* to obtain a new cursor.

is_other()

Check if the union tag is `other`.

Return type bool

is_reset()

Check if the union tag is `reset`.

Return type bool

other = ListFolderLongpollError('other', None)

reset = ListFolderLongpollError('reset', None)

class `dropbox.files.`**`ListFolderLongpollResult`**(*changes=None*, *backoff=None*)

Bases: `object`

Variables

- ***changes*** – Indicates whether new changes are available. If true, call *dropbox.dropbox.Dropbox.files_list_folder_continue()* to retrieve the changes.
- ***backoff*** – If present, backoff for at least this many seconds before calling *dropbox.dropbox.Dropbox.files_list_folder_longpoll()* again.

backoff

If present, backoff for at least this many seconds before calling *dropbox.dropbox.Dropbox.files_list_folder_longpoll()* again.

Return type long

changes
Indicates whether new changes are available. If true, call `dropbox.dropbox.Dropbox.files_list_folder_continue()` to retrieve the changes.

Return type bool

class `dropbox.files.`**`ListFolderResult`**(*entries=None*, *cursor=None*, *has_more=None*)
Bases: `object`

Variables

- ***entries*** – The files and (direct) subfolders in the folder.
- ***cursor*** – Pass the cursor into `dropbox.dropbox.Dropbox.files_list_folder_continue()` to see what's changed in the folder since your previous query.
- ***has_more*** – If true, then there are more entries available. Pass the cursor to `dropbox.dropbox.Dropbox.files_list_folder_continue()` to retrieve the rest.

cursor
Pass the cursor into `dropbox.dropbox.Dropbox.files_list_folder_continue()` to see what's changed in the folder since your previous query.

Return type str

entries
The files and (direct) subfolders in the folder.

Return type list of [Metadata]

has_more
If true, then there are more entries available. Pass the cursor to `dropbox.dropbox.Dropbox.files_list_folder_continue()` to retrieve the rest.

Return type bool

class `dropbox.files.`**`ListRevisionsArg`**(*path=None*, *limit=None*)
Bases: `object`

Variables

- ***path*** – The path to the file you want to see the revisions of.
- ***limit*** – The maximum number of revision entries returned.

limit
The maximum number of revision entries returned.

Return type long

path
The path to the file you want to see the revisions of.

Return type str

class `dropbox.files.`**`ListRevisionsError`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

get_path()
Only call this if `is_path()` is true.

Return type *LookupError*

is_other()
Check if the union tag is `other`.

Return type bool

is_path()
Check if the union tag is `path`.

Return type bool

other = ListRevisionsError('other', None)

classmethod path(*val*)
Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (`LookupError`) –

Return type *ListRevisionsError*

class dropbox.files.**ListRevisionsResult**(*is_deleted=None*, *entries=None*)
Bases: `object`

Variables

- ***is_deleted*** – If the file is deleted.
- ***entries*** – The revisions for the file. Only non-delete revisions will show up here.

entries
The revisions for the file. Only non-delete revisions will show up here.

Return type list of [FileMetadata]

is_deleted
If the file is deleted.

Return type bool

class dropbox.files.**LookUpPropertiesError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***property_group_not_found*** – This property group does not exist for this file.

is_property_group_not_found()
Check if the union tag is `property_group_not_found`.

Return type bool

property_group_not_found = LookUpPropertiesError('property_group_not_found', None)

class dropbox.files.**LookupError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***not_found*** – There is nothing at the given path.
- ***not_file*** – We were expecting a file, but the given path refers to something that isn't a file.

- ***not_folder*** – We were expecting a folder, but the given path refers to something that isn't a folder.
- ***restricted_content*** – The file cannot be transferred because the content is restricted. For example, sometimes there are legal restrictions due to copyright claims.
- ***invalid_path_root*** (`PathRootError`) – The path root parameter provided is invalid.

get_invalid_path_root()

The path root parameter provided is invalid.

Only call this if `is_invalid_path_root()` is true.

Return type *PathRootError*

get_malformed_path()

Only call this if `is_malformed_path()` is true.

Return type Optional[str]

classmethod invalid_path_root(*val*)

Create an instance of this class set to the `invalid_path_root` tag with value `val`.

Parameters **val** (`PathRootError`) –

Return type *LookupError*

is_invalid_path_root()

Check if the union tag is `invalid_path_root`.

Return type bool

is_malformed_path()

Check if the union tag is `malformed_path`.

Return type bool

is_not_file()

Check if the union tag is `not_file`.

Return type bool

is_not_folder()

Check if the union tag is `not_folder`.

Return type bool

is_not_found()

Check if the union tag is `not_found`.

Return type bool

is_other()

Check if the union tag is `other`.

Return type bool

is_restricted_content()

Check if the union tag is `restricted_content`.

Return type bool

classmethod malformed_path(*val*)

Create an instance of this class set to the `malformed_path` tag with value `val`.

Parameters **val** (`Optional[str]`) –

Return type *LookupError*

not_file = LookupError('not_file', None)

not_folder = LookupError('not_folder', None)

not_found = LookupError('not_found', None)

other = LookupError('other', None)

restricted_content = Lookup Error('restricted_content', None)

class dropbox.files.**MediaInfo**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables

- ***pending*** – Indicate the photo/video is still under processing and metadata is not available yet.
- ***metadata*** (MediaMetadata) – The metadata for the photo/video.

get_metadata()
The metadata for the photo/video.

Only call this if *is_metadata()* is true.

Return type *MediaMetadata*

is_metadata()
Check if the union tag is metadata.

Return type bool

is_pending()
Check if the union tag is pending.

Return type bool

classmethod metadata(*val*)
Create an instance of this class set to the metadata tag with value val.

Parameters **val** (MediaMetadata) –

Return type *MediaInfo*

pending = MediaInfo('pending', None)

class dropbox.files.**MediaMetadata**(*dimensions=None*, *location=None*, *time_taken=None*)
Bases: object

Metadata for a photo or video.

Variables

- ***dimensions*** – Dimension of the photo/video.
- ***location*** – The GPS coordinate of the photo/video.
- ***time_taken*** – The timestamp when the photo/video is taken.

dimensions
Dimension of the photo/video.

Return type *Dimensions*

location
The GPS coordinate of the photo/video.

Return type *GpsCoordinates*

time_taken
The timestamp when the photo/video is taken.

Return type datetime.datetime

class `dropbox.files.`**`Metadata`**(*name=None*, *path_lower=None*, *path_display=None*, *parent_shared_folder_id=None*)

Bases: `object`

Metadata for a file or folder.

Variables

- ***name*** – The last component of the path (including extension). This never contains a slash.
- ***path_lower*** – The lowercased full path in the user's Dropbox. This always starts with a slash. This field will be null if the file or folder is not mounted.
- ***path_display*** – The cased path to be used for display purposes only. In rare instances the casing will not correctly match the user's filesystem, but this behavior will match the path provided in the Core API v1. Changes to the casing of paths won't be returned by `dropbox.dropbox.Dropbox.files_list_folder_continue()`. This field will be null if the file or folder is not mounted.
- ***parent_shared_folder_id*** – Deprecated. Please use `FileSharingInfo.parent_shared_folder_id` or `FolderSharingInfo.parent_shared_folder_id` instead.

name
The last component of the path (including extension). This never contains a slash.

Return type str

parent_shared_folder_id
Deprecated. Please use `FileSharingInfo.parent_shared_folder_id` or `FolderSharingInfo.parent_shared_folder_id` instead.

Return type str

path_display
The cased path to be used for display purposes only. In rare instances the casing will not correctly match the user's filesystem, but this behavior will match the path provided in the Core API v1. Changes to the casing of paths won't be returned by `dropbox.dropbox.Dropbox.files_list_folder_continue()`. This field will be null if the file or folder is not mounted.

Return type str

path_lower
The lowercased full path in the user's Dropbox. This always starts with a slash. This field will be null if the file or folder is not mounted.

Return type str

class `dropbox.files.`**`PathRootError`**(*path_root=None*)

Bases: `object`

Variables ***path_root*** – The user's latest path root value. None if the user no longer has a path root.

path_root
The user's latest path root value. None if the user no longer has a path root.

Return type str

class dropbox.files.**PhotoMetadata**(*dimensions=None*, *location=None*, *time_taken=None*)
Bases: *dropbox.files.MediaMetadata*

Metadata for a photo.

class dropbox.files.**PreviewArg**(*path=None*, *rev=None*)
Bases: object

Variables

- ***path*** – The path of the file to preview.
- ***rev*** – Deprecated. Please specify revision in path instead.

path
The path of the file to preview.

Return type str

rev
Deprecated. Please specify revision in path instead.

Return type str

class dropbox.files.**PreviewError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables

- ***path*** (LookupError) – An error occurs when downloading metadata for the file.
- ***in_progress*** – This preview generation is still in progress and the file is not ready for preview yet.
- ***unsupported_extension*** – The file extension is not supported preview generation.
- ***unsupported_content*** – The file content is not supported for preview generation.

get_path()
An error occurs when downloading metadata for the file.

Only call this if *is_path()* is true.

Return type *LookupError*

in_progress = PreviewError('in_progress', None)

is_in_progress()
Check if the union tag is in_progress.

Return type bool

is_path()
Check if the union tag is path.

Return type bool

is_unsupported_content()
Check if the union tag is unsupported_content.

Return type bool

is_unsupported_extension()
Check if the union tag is `unsupported_extension`.

Return type bool

classmethod path(*val*)
Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (`LookupError`) –

Return type *PreviewError*

unsupported_content = PreviewError('unsupported_content', None)

unsupported_extension = PreviewError('unsupported_extension', None)

class `dropbox.files.`**PropertiesError**(*tag*, *value=None*)
Bases: `dropbox.properties.PropertyTemplateError`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

get_path()
Only call this if *is_path()* is true.

Return type *LookupError*

is_path()
Check if the union tag is `path`.

Return type bool

classmethod path(*val*)
Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (`LookupError`) –

Return type *PropertiesError*

class `dropbox.files.`**PropertyGroupUpdate**(*template_id=None*, *add_or_update_fields=None*, *remove_fields=None*)
Bases: `object`

Variables

- ***template_id*** – A unique identifier for a property template.
- ***add_or_update_fields*** – List of property fields to update if the field already exists. If the field doesn't exist, add the field to the property group.
- ***remove_fields*** – List of property field names to remove from property group if the field exists.

add_or_update_fields
List of property fields to update if the field already exists. If the field doesn't exist, add the field to the property group.

Return type list of [properties.PropertyField_validator]

remove_fields
List of property field names to remove from property group if the field exists.

Return type list of [str]

template_id
A unique identifier for a property template.

Return type str

class dropbox.files.**PropertyGroupWithPath**(*path=None*, *property_groups=None*)
Bases: object

Variables

- ***path*** – A unique identifier for the file.
- ***property_groups*** – Filled custom property templates associated with a file.

path
A unique identifier for the file.

Return type str

property_groups
Filled custom property templates associated with a file.

Return type list of [properties.PropertyGroup_validator]

class dropbox.files.**RelocationArg**(*from_path=None*, *to_path=None*, *allow_shared_folder=None*, *autorename=None*)
Bases: *dropbox.files.RelocationPath*

Variables

- ***allow_shared_folder*** – If true, *dropbox.dropbox.Dropbox.files_copy()* will copy contents in shared folder, otherwise RelocationError.cant_copy_shared_folder will be returned if from_path contains shared folder. This field is always true for *dropbox.dropbox.Dropbox.files_move()*.
- ***autorename*** – If there's a conflict, have the Dropbox server try to autorename the file to avoid the conflict.

allow_shared_folder
If true, *dropbox.dropbox.Dropbox.files_copy()* will copy contents in shared folder, otherwise RelocationError.cant_copy_shared_folder will be returned if from_path contains shared folder. This field is always true for *dropbox.dropbox.Dropbox.files_move()*.

Return type bool

autorename
If there's a conflict, have the Dropbox server try to autorename the file to avoid the conflict.

Return type bool

class dropbox.files.**RelocationBatchArg**(*entries=None*, *allow_shared_folder=None*, *autorename=None*)
Bases: object

Variables

- ***entries*** – List of entries to be moved or copied. Each entry is *RelocationPath*.
- ***allow_shared_folder*** – If true, *dropbox.dropbox.Dropbox.files_copy_batch()* will copy contents in shared folder, otherwise RelocationError.cant_copy_shared_folder will be returned if RelocationPath.from_path contains shared folder. This field is always true for *dropbox.dropbox.Dropbox.files_move_batch()*.
- ***autorename*** – If there's a conflict with any file, have the Dropbox server try to autorename that file to avoid the conflict.

allow_shared_folder
If true, *dropbox.dropbox.Dropbox.files_copy_batch()* will copy contents in shared folder, otherwise `RelocationError.cant_copy_shared_folder` will be returned if `RelocationPath.from_path` contains shared folder. This field is always true for *dropbox.dropbox.Dropbox.files_move_batch()*.

Return type bool

autorename
If there's a conflict with any file, have the Dropbox server try to autorename that file to avoid the conflict.

Return type bool

entries
List of entries to be moved or copied. Each entry is *RelocationPath*.

Return type list of [RelocationPath]

class `dropbox.files.`**RelocationBatchError**(*tag*, *value=None*)
Bases: *dropbox.files.RelocationError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***duplicated_or_nested_paths*** – There are duplicated/nested paths among `RelocationArg.from_path` and `RelocationArg.to_path`.
- ***too_many_write_operations*** – There are too many write operations in user's Dropbox. Please retry this request.

duplicated_or_nested_paths = RelocationBatchError('duplicated_or_nested_paths', None)

is_duplicated_or_nested_paths()
Check if the union tag is `duplicated_or_nested_paths`.

Return type bool

is_too_many_write_operations()
Check if the union tag is `too_many_write_operations`.

Return type bool

too_many_write_operations = RelocationBatchError('too_many_write_operations', None)

class `dropbox.files.`**RelocationBatchJobStatus**(*tag*, *value=None*)
Bases: *dropbox.async.PollResultBase*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***complete*** (`RelocationBatchResult`) – The copy or move batch job has finished.
- ***failed*** (`RelocationBatchError`) – The copy or move batch job has failed with exception.

classmethod complete(*val*)
Create an instance of this class set to the `complete` tag with value `val`.

Parameters **val** (`RelocationBatchResult`) –

Return type *RelocationBatchJobStatus*

classmethod failed(*val*)
Create an instance of this class set to the `failed` tag with value `val`.

Parameters **val** (`RelocationBatchError`) –

Return type *RelocationBatchJobStatus*

get_complete()
The copy or move batch job has finished.

Only call this if *`is_complete()`* is true.

Return type *RelocationBatchResult*

get_failed()
The copy or move batch job has failed with exception.

Only call this if *`is_failed()`* is true.

Return type *RelocationBatchError*

is_complete()
Check if the union tag is `complete`.

Return type bool

is_failed()
Check if the union tag is `failed`.

Return type bool

class `dropbox.files.`**RelocationBatchResult**(*entries=None*)
Bases: `object`

entries

Return type list of [RelocationResult]

class `dropbox.files.`**RelocationError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***cant_copy_shared_folder*** – Shared folders can't be copied.
- ***cant_nest_shared_folder*** – Your move operation would result in nested shared folders. This is not allowed.
- ***cant_move_folder_into_itself*** – You cannot move a folder into itself.
- ***too_many_files*** – The operation would involve more than 10,000 files and folders.

cant_copy_shared_folder = RelocationError('cant_copy_shared_folder', None)

cant_move_folder_into_itself = RelocationError('cant_move_folder_into_itself', None)

cant_nest_shared_folder = RelocationError('cant_nest_shared_folder', None)

classmethod from_lookup(*val*)
Create an instance of this class set to the `from_lookup` tag with value `val`.

Parameters **val** (`LookupError`) –

Return type *RelocationError*

classmethod from_write(*val*)
Create an instance of this class set to the `from_write` tag with value `val`.

Parameters **val** (`WriteError`) –

Return type *RelocationError*

get_from_lookup()
Only call this if *`is_from_lookup()`* is true.

Return type *LookupError*

get_from_write()
Only call this if *`is_from_write()`* is true.

Return type *WriteError*

get_to()
Only call this if *`is_to()`* is true.

Return type *WriteError*

is_cant_copy_shared_folder()
Check if the union tag is `cant_copy_shared_folder`.

Return type bool

is_cant_move_folder_into_itself()
Check if the union tag is `cant_move_folder_into_itself`.

Return type bool

is_cant_nest_shared_folder()
Check if the union tag is `cant_nest_shared_folder`.

Return type bool

is_from_lookup()
Check if the union tag is `from_lookup`.

Return type bool

is_from_write()
Check if the union tag is `from_write`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_to()
Check if the union tag is `to`.

Return type bool

is_too_many_files()
Check if the union tag is `too_many_files`.

Return type bool

other = RelocationError('other', None)

classmethod to(*val*)
Create an instance of this class set to the `to` tag with value `val`.

Parameters **val** (`WriteError`) –

Return type *RelocationError*

too_many_files = RelocationError('too_many_files', None)

class `dropbox.files.`**`RelocationPath`**(*from_path=None*, *to_path=None*)
Bases: `object`

Variables

- ***from_path*** – Path in the user's Dropbox to be copied or moved.
- ***to_path*** – Path in the user's Dropbox that is the destination.

from_path
Path in the user's Dropbox to be copied or moved.

Return type str

to_path
Path in the user's Dropbox that is the destination.

Return type str

class `dropbox.files.`**`RelocationResult`**(*metadata=None*)
Bases: `object`

metadata

Return type *Metadata*

class `dropbox.files.`**`RemovePropertiesArg`**(*path=None*, *property_template_ids=None*)
Bases: `object`

Variables

- ***path*** – A unique identifier for the file.
- ***property_template_ids*** – A list of identifiers for a property template created by route properties/template/add.

path
A unique identifier for the file.

Return type str

property_template_ids
A list of identifiers for a property template created by route properties/template/add.

Return type list of [str]

class `dropbox.files.`**`RemovePropertiesError`**(*tag*, *value=None*)
Bases: *dropbox.files.PropertiesError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

get_property_group_lookup()
Only call this if *is_property_group_lookup()* is true.

Return type *LookUpPropertiesError*

is_property_group_lookup()
Check if the union tag is `property_group_lookup`.

Return type bool

classmethod property_group_lookup(*val*)
Create an instance of this class set to the `property_group_lookup` tag with value `val`.

Parameters **val** (`LookUpPropertiesError`) –

Return type *RemovePropertiesError*

class `dropbox.files.`**RestoreArg**(*path=None*, *rev=None*)
Bases: `object`

Variables

- ***path*** – The path to the file you want to restore.
- ***rev*** – The revision to restore for the file.

path
The path to the file you want to restore.

Return type str

rev
The revision to restore for the file.

Return type str

class `dropbox.files.`**RestoreError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***path_lookup*** (`LookupError`) – An error occurs when downloading metadata for the file.
- ***path_write*** (`WriteError`) – An error occurs when trying to restore the file to that path.
- ***invalid_revision*** – The revision is invalid. It may point to a different file.

get_path_lookup()
An error occurs when downloading metadata for the file.

Only call this if *`is_path_lookup()`* is true.

Return type *LookupError*

get_path_write()
An error occurs when trying to restore the file to that path.

Only call this if *`is_path_write()`* is true.

Return type *WriteError*

invalid_revision = RestoreError('invalid_revision', None)

is_invalid_revision()
Check if the union tag is `invalid_revision`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_path_lookup()
Check if the union tag is `path_lookup`.

Return type bool

is_path_write()
Check if the union tag is `path_write`.

Return type bool

other = RestoreError('other', None)

classmethod path_lookup(*val*)
Create an instance of this class set to the `path_lookup` tag with value `val`.

Parameters **val** (`LookupError`) –

Return type *RestoreError*

classmethod path_write(*val*)
Create an instance of this class set to the `path_write` tag with value `val`.

Parameters **val** (`WriteError`) –

Return type *RestoreError*

class `dropbox.files.`**SaveCopyReferenceArg**(*copy_reference=None*, *path=None*)
Bases: `object`

Variables

- ***copy_reference*** – A copy reference returned by *dropbox.dropbox.Dropbox.files_copy_reference_get()*.
- ***path*** – Path in the user's Dropbox that is the destination.

copy_reference
A copy reference returned by *dropbox.dropbox.Dropbox.files_copy_reference_get()*.

Return type str

path
Path in the user's Dropbox that is the destination.

Return type str

class `dropbox.files.`**SaveCopyReferenceError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***invalid_copy_reference*** – The copy reference is invalid.
- ***no_permission*** – You don't have permission to save the given copy reference. Please make sure this app is same app which created the copy reference and the source user is still linked to the app.
- ***not_found*** – The file referenced by the copy reference cannot be found.
- ***too_many_files*** – The operation would involve more than 10,000 files and folders.

get_path()
Only call this if *is_path()* is true.

Return type *WriteError*

invalid_copy_reference = SaveCopyReferenceError('invalid_copy_reference', None)

is_invalid_copy_reference()
Check if the union tag is `invalid_copy_reference`.

Return type bool

is_no_permission()
Check if the union tag is `no_permission`.

Return type bool

is_not_found()
Check if the union tag is `not_found`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_path()
Check if the union tag is `path`.

Return type bool

is_too_many_files()
Check if the union tag is `too_many_files`.

Return type bool

no_permission = SaveCopyReferenceError('no_permission', None)

not_found = SaveCopyReferenceError('not_found', None)

other = SaveCopyReferenceError('other', None)

classmethod path(*val*)
Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (`WriteError`) –

Return type *SaveCopyReferenceError*

too_many_files = SaveCopyReferenceError('too_many_files', None)

class dropbox.files.**SaveCopyReferenceResult**(*metadata=None*)
Bases: `object`

Variables ***metadata*** – The metadata of the saved file or folder in the user's Dropbox.

metadata
The metadata of the saved file or folder in the user's Dropbox.

Return type *Metadata*

class dropbox.files.**SaveUrlArg**(*path=None*, *url=None*)
Bases: `object`

Variables

- ***path*** – The path in Dropbox where the URL will be saved to.
- ***url*** – The URL to be saved.

path
The path in Dropbox where the URL will be saved to.

Return type str

url
The URL to be saved.

Return type str

class `dropbox.files.`**`SaveUrlError`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***download_failed*** – Failed downloading the given URL.
- ***invalid_url*** – The given URL is invalid.
- ***not_found*** – The file where the URL is saved to no longer exists.

download_failed = SaveUrlError('download_failed', None)

get_path()
Only call this if *is_path()* is true.

Return type *WriteError*

invalid_url = SaveUrlError('invalid_url', None)

is_download_failed()
Check if the union tag is `download_failed`.

Return type bool

is_invalid_url()
Check if the union tag is `invalid_url`.

Return type bool

is_not_found()
Check if the union tag is `not_found`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_path()
Check if the union tag is `path`.

Return type bool

not_found = SaveUrlError('not_found', None)

other = SaveUrlError('other', None)

classmethod path(*val*)
Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (`WriteError`) –

Return type *SaveUrlError*

class `dropbox.files.`**`SaveUrlJobStatus`**(*tag*, *value=None*)
Bases: *dropbox.async.PollResultBase*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***complete*** (`FileMetadata`) – Metadata of the file where the URL is saved to.

classmethod **`complete`**(*val*)
Create an instance of this class set to the `complete` tag with value `val`.

Parameters **`val`** (`FileMetadata`) –

Return type *SaveUrlJobStatus*

classmethod **`failed`**(*val*)
Create an instance of this class set to the `failed` tag with value `val`.

Parameters **`val`** (`SaveUrlError`) –

Return type *SaveUrlJobStatus*

`get_complete`()
Metadata of the file where the URL is saved to.

Only call this if *`is_complete()`* is true.

Return type *FileMetadata*

`get_failed`()
Only call this if *`is_failed()`* is true.

Return type *SaveUrlError*

`is_complete`()
Check if the union tag is `complete`.

Return type bool

`is_failed`()
Check if the union tag is `failed`.

Return type bool

class `dropbox.files.`**`SaveUrlResult`**(*tag*, *value=None*)
Bases: *dropbox.async.LaunchResultBase*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***complete*** (`FileMetadata`) – Metadata of the file where the URL is saved to.

classmethod **`complete`**(*val*)
Create an instance of this class set to the `complete` tag with value `val`.

Parameters **`val`** (`FileMetadata`) –

Return type *SaveUrlResult*

`get_complete`()
Metadata of the file where the URL is saved to.

Only call this if *`is_complete()`* is true.

Return type *FileMetadata*

is_complete()
Check if the union tag is `complete`.

Return type bool

class dropbox.files.**SearchArg**(*path=None*, *query=None*, *start=None*, *max_results=None*, *mode=None*)
Bases: `object`

Variables

- ***path*** – The path in the user's Dropbox to search. Should probably be a folder.
- ***query*** – The string to search for. The search string is split on spaces into multiple tokens. For file name searching, the last token is used for prefix matching (i.e. "bat c" matches "bat cave" but not "batman car").
- ***start*** – The starting index within the search results (used for paging).
- ***max_results*** – The maximum number of search results to return.
- ***mode*** – The search mode (filename, filename_and_content, or deleted_filename). Note that searching file content is only available for Dropbox Business accounts.

max_results
The maximum number of search results to return.

Return type long

mode
The search mode (filename, filename_and_content, or deleted_filename). Note that searching file content is only available for Dropbox Business accounts.

Return type *SearchMode*

path
The path in the user's Dropbox to search. Should probably be a folder.

Return type str

query
The string to search for. The search string is split on spaces into multiple tokens. For file name searching, the last token is used for prefix matching (i.e. "bat c" matches "bat cave" but not "batman car").

Return type str

start
The starting index within the search results (used for paging).

Return type long

class dropbox.files.**SearchError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

get_path()
Only call this if *`is_path()`* is true.

Return type *LookupError*

is_other()
Check if the union tag is `other`.

Return type bool

is_path()
Check if the union tag is `path`.

Return type bool

other = SearchError('other', None)

classmethod path(*val*)
Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (`LookupError`) –

Return type *SearchError*

class dropbox.files.**SearchMatch**(*match_type=None*, *metadata=None*)
Bases: `object`

Variables

- ***match_type*** – The type of the match.
- ***metadata*** – The metadata for the matched file or folder.

match_type
The type of the match.

Return type *SearchMatchType*

metadata
The metadata for the matched file or folder.

Return type *Metadata*

class dropbox.files.**SearchMatchType**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Indicates what type of match was found for a given item.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***filename*** – This item was matched on its file or folder name.
- ***content*** – This item was matched based on its file contents.
- ***both*** – This item was matched based on both its contents and its file name.

both = SearchMatchType('both', None)

content = SearchMatchType('content', None)

filename = SearchMatchType('filename', None)

is_both()
Check if the union tag is `both`.

Return type bool

is_content()
Check if the union tag is `content`.

Return type bool

is_filename()
Check if the union tag is `filename`.

Return type bool

class `dropbox.files.`**`SearchMode`**(*tag*, *value=None*)

Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***filename*** – Search file and folder names.
- ***filename_and_content*** – Search file and folder names as well as file contents.
- ***deleted_filename*** – Search for deleted file and folder names.

`deleted_filename` = SearchMode('deleted_filename', None)

`filename` = SearchMode('filename', None)

`filename_and_content` = SearchMode('filename_and_content', None)

`is_deleted_filename`()

Check if the union tag is `deleted_filename`.

Return type bool

`is_filename`()

Check if the union tag is `filename`.

Return type bool

`is_filename_and_content`()

Check if the union tag is `filename_and_content`.

Return type bool

class `dropbox.files.`**`SearchResult`**(*matches=None*, *more=None*, *start=None*)

Bases: `object`

Variables

- ***matches*** – A list (possibly empty) of matches for the query.
- ***more*** – Used for paging. If true, indicates there is another page of results available that can be fetched by calling *`dropbox.dropbox.Dropbox.files_search()`* again.
- ***start*** – Used for paging. Value to set the start argument to when calling *`dropbox.dropbox.Dropbox.files_search()`* to fetch the next page of results.

`matches`

A list (possibly empty) of matches for the query.

Return type list of [SearchMatch]

`more`

Used for paging. If true, indicates there is another page of results available that can be fetched by calling *`dropbox.dropbox.Dropbox.files_search()`* again.

Return type bool

`start`

Used for paging. Value to set the start argument to when calling *`dropbox.dropbox.Dropbox.files_search()`* to fetch the next page of results.

Return type long

class `dropbox.files.`**`SharingInfo`**(*read_only=None*)
Bases: `object`

Sharing info for a file or folder.

Variables ***read_only*** – True if the file or folder is inside a read-only shared folder.

`read_only`
True if the file or folder is inside a read-only shared folder.

Return type bool

class `dropbox.files.`**`ThumbnailArg`**(*path=None*, *format=None*, *size=None*)
Bases: `object`

Variables

- ***path*** – The path to the image file you want to thumbnail.
- ***format*** – The format for the thumbnail image, jpeg (default) or png. For images that are photos, jpeg should be preferred, while png is better for screenshots and digital arts.
- ***size*** – The size for the thumbnail image.

`format`
The format for the thumbnail image, jpeg (default) or png. For images that are photos, jpeg should be preferred, while png is better for screenshots and digital arts.

Return type *ThumbnailFormat*

`path`
The path to the image file you want to thumbnail.

Return type str

`size`
The size for the thumbnail image.

Return type *ThumbnailSize*

class `dropbox.files.`**`ThumbnailError`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***path*** (`LookupError`) – An error occurs when downloading metadata for the image.
- ***unsupported_extension*** – The file extension doesn't allow conversion to a thumbnail.
- ***unsupported_image*** – The image cannot be converted to a thumbnail.
- ***conversion_error*** – An error occurs during thumbnail conversion.

`conversion_error` = ThumbnailError('conversion_error', None)

`get_path`()
An error occurs when downloading metadata for the image.

Only call this if *`is_path()`* is true.

Return type *LookupError*

is_conversion_error()
Check if the union tag is `conversion_error`.

Return type bool

is_path()
Check if the union tag is `path`.

Return type bool

is_unsupported_extension()
Check if the union tag is `unsupported_extension`.

Return type bool

is_unsupported_image()
Check if the union tag is `unsupported_image`.

Return type bool

classmethod path(*val*)
Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (`LookupError`) –

Return type *ThumbnailError*

unsupported_extension = ThumbnailError('unsupported_extension', None)

unsupported_image = ThumbnailError('unsupported_image', None)

class dropbox.files.**ThumbnailFormat**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

is_jpeg()
Check if the union tag is `jpeg`.

Return type bool

is_png()
Check if the union tag is `png`.

Return type bool

jpeg = ThumbnailFormat('jpeg', None)

png = ThumbnailFormat('png', None)

class dropbox.files.**ThumbnailSize**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***w32h32*** – 32 by 32 px.
- ***w64h64*** – 64 by 64 px.
- ***w128h128*** – 128 by 128 px.
- ***w640h480*** – 640 by 480 px.
- ***w1024h768*** – 1024 by 768.

is_w1024h768()
Check if the union tag is `w1024h768`.

Return type bool

is_w128h128()
Check if the union tag is `w128h128`.

Return type bool

is_w32h32()
Check if the union tag is `w32h32`.

Return type bool

is_w640h480()
Check if the union tag is `w640h480`.

Return type bool

is_w64h64()
Check if the union tag is `w64h64`.

Return type bool

w1024h768 = ThumbnailSize('w1024h768', None)

w128h128 = ThumbnailSize('w128h128', None)

w32h32 = ThumbnailSize('w32h32', None)

w640h480 = ThumbnailSize('w640h480', None)

w64h64 = ThumbnailSize('w64h64', None)

class dropbox.files.**UpdatePropertiesError**(*tag*, *value=None*)
Bases: *dropbox.files.InvalidPropertyGroupError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

get_property_group_lookup()
Only call this if *is_property_group_lookup()* is true.

Return type *LookUpPropertiesError*

is_property_group_lookup()
Check if the union tag is `property_group_lookup`.

Return type bool

classmethod property_group_lookup(*val*)
Create an instance of this class set to the `property_group_lookup` tag with value `val`.

Parameters **val** (`LookUpPropertiesError`) –

Return type *UpdatePropertiesError*

class dropbox.files.**UpdatePropertyGroupArg**(*path=None*, *update_property_groups=None*)
Bases: `object`

Variables

- ***path*** – A unique identifier for the file.
- ***update_property_groups*** – Filled custom property templates associated with a file.

path
A unique identifier for the file.

Return type str

update_property_groups
Filled custom property templates associated with a file.

Return type list of [PropertyGroupUpdate]

class dropbox.files.**UploadError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables ***path*** (UploadWriteFailed) – Unable to save the uploaded contents to a file.

get_path()
Unable to save the uploaded contents to a file.

Only call this if *is_path()* is true.

Return type *UploadWriteFailed*

is_other()
Check if the union tag is other.

Return type bool

is_path()
Check if the union tag is path.

Return type bool

other = UploadError('other', None)

classmethod path(*val*)
Create an instance of this class set to the path tag with value val.

Parameters **val** (UploadWriteFailed) –

Return type *UploadError*

class dropbox.files.**UploadErrorWithProperties**(*tag*, *value=None*)
Bases: *dropbox.files.UploadError*

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

get_properties_error()
Only call this if *is_properties_error()* is true.

Return type *InvalidPropertyGroupError*

is_properties_error()
Check if the union tag is properties_error.

Return type bool

classmethod properties_error(*val*)
Create an instance of this class set to the properties_error tag with value val.

Parameters **val** (InvalidPropertyGroupError) –

Return type *UploadErrorWithProperties*

class dropbox.files.**UploadSessionAppendArg**(*cursor=None*, *close=None*)
Bases: object

Variables

- ***cursor*** – Contains the upload session ID and the offset.
- ***close*** – If true, the current session will be closed, at which point you won't be able to call *dropbox.dropbox.Dropbox.files_upload_session_append_v2()* anymore with the current session.

close
If true, the current session will be closed, at which point you won't be able to call *dropbox.dropbox.Dropbox.files_upload_session_append_v2()* anymore with the current session.

Return type bool

cursor
Contains the upload session ID and the offset.

Return type *UploadSessionCursor*

class dropbox.files.**UploadSessionCursor**(*session_id=None*, *offset=None*)
Bases: object

Variables

- ***session_id*** – The upload session ID (returned by *dropbox.dropbox.Dropbox.files_upload_session_start()*).
- ***offset*** – The amount of data that has been uploaded so far. We use this to make sure upload data isn't lost or duplicated in the event of a network error.

offset
The amount of data that has been uploaded so far. We use this to make sure upload data isn't lost or duplicated in the event of a network error.

Return type long

session_id
The upload session ID (returned by *dropbox.dropbox.Dropbox.files_upload_session_start()*).

Return type str

class dropbox.files.**UploadSessionFinishArg**(*cursor=None*, *commit=None*)
Bases: object

Variables

- ***cursor*** – Contains the upload session ID and the offset.
- ***commit*** – Contains the path and other optional modifiers for the commit.

commit
Contains the path and other optional modifiers for the commit.

Return type *CommitInfo*

cursor
Contains the upload session ID and the offset.

Return type *UploadSessionCursor*

class dropbox.files.**UploadSessionFinishBatchArg**(*entries=None*)
Bases: object

Variables ***entries*** – Commit information for each file in the batch.

entries
Commit information for each file in the batch.

Return type list of [UploadSessionFinishArg]

class dropbox.files.**UploadSessionFinishBatchJobStatus**(*tag*, *value=None*)
Bases: *dropbox.async.PollResultBase*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***complete*** (`UploadSessionFinishBatchResult`) – The *dropbox.dropbox.Dropbox.files_upload_session_finish_batch()* has finished.

classmethod complete(*val*)
Create an instance of this class set to the `complete` tag with value `val`.

Parameters **val** (`UploadSessionFinishBatchResult`) –

Return type *UploadSessionFinishBatchJobStatus*

get_complete()
The *dropbox.dropbox.Dropbox.files_upload_session_finish_batch()* has finished.

Only call this if *is_complete()* is true.

Return type *UploadSessionFinishBatchResult*

is_complete()
Check if the union tag is `complete`.

Return type bool

class dropbox.files.**UploadSessionFinishBatchResult**(*entries=None*)
Bases: `object`

Variables ***entries*** – Commit result for each file in the batch.

entries
Commit result for each file in the batch.

Return type list of [UploadSessionFinishBatchResultEntry]

class dropbox.files.**UploadSessionFinishBatchResultEntry**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

classmethod failure(*val*)
Create an instance of this class set to the `failure` tag with value `val`.

Parameters **val** (`UploadSessionFinishError`) –

Return type *UploadSessionFinishBatchResultEntry*

get_failure()
Only call this if *is_failure()* is true.

Return type *UploadSessionFinishError*

get_success()
Only call this if *is_success()* is true.

Return type *FileMetadata*

is_failure()
Check if the union tag is `failure`.

Return type bool

is_success()
Check if the union tag is `success`.

Return type bool

classmethod success(*val*)
Create an instance of this class set to the `success` tag with value `val`.

Parameters **val** (`FileMetadata`) –

Return type *UploadSessionFinishBatchResultEntry*

class dropbox.files.**UploadSessionFinishError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***lookup_failed*** (`UploadSessionLookupError`) – The session arguments are incorrect; the value explains the reason.
- ***path*** (`WriteError`) – Unable to save the uploaded contents to a file.
- ***too_many_shared_folder_targets*** – The batch request commits files into too many different shared folders. Please limit your batch request to files contained in a single shared folder.

get_lookup_failed()
The session arguments are incorrect; the value explains the reason.

Only call this if *`is_lookup_failed()`* is true.

Return type *UploadSessionLookupError*

get_path()
Unable to save the uploaded contents to a file.

Only call this if *`is_path()`* is true.

Return type *WriteError*

is_lookup_failed()
Check if the union tag is `lookup_failed`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_path()
Check if the union tag is `path`.

Return type bool

is_too_many_shared_folder_targets()
Check if the union tag is `too_many_shared_folder_targets`.

Return type bool

classmethod lookup_failed(*val*)

Create an instance of this class set to the `lookup_failed` tag with value `val`.

Parameters **val** (`UploadSessionLookupError`) –

Return type *UploadSessionFinishError*

other = UploadSessionFinishError('other', None)

classmethod path(*val*)

Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (`WriteError`) –

Return type *UploadSessionFinishError*

too_many_shared_folder_targets = UploadSessionFinishError('too_many_shared_folder_targets', None)

class `dropbox.files.`**UploadSessionLookupError**(*tag*, *value=None*)

Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***not_found*** – The upload session id was not found.
- ***incorrect_offset*** (`UploadSessionOffsetError`) – The specified offset was incorrect. See the value for the correct offset. This error may occur when a previous request was received and processed successfully but the client did not receive the response, e.g. due to a network error.
- ***closed*** – You are attempting to append data to an upload session that has alread been closed (i.e. committed).
- ***not_closed*** – The session must be closed before calling upload_session/finish_batch.

closed = UploadSessionLookupError('closed', None)

get_incorrect_offset()

The specified offset was incorrect. See the value for the correct offset. This error may occur when a previous request was received and processed successfully but the client did not receive the response, e.g. due to a network error.

Only call this if *`is_incorrect_offset()`* is true.

Return type *UploadSessionOffsetError*

classmethod incorrect_offset(*val*)

Create an instance of this class set to the `incorrect_offset` tag with value `val`.

Parameters **val** (`UploadSessionOffsetError`) –

Return type *UploadSessionLookupError*

is_closed()

Check if the union tag is `closed`.

Return type bool

is_incorrect_offset()

Check if the union tag is `incorrect_offset`.

Return type bool

is_not_closed()
Check if the union tag is `not_closed`.

Return type bool

is_not_found()
Check if the union tag is `not_found`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

not_closed = UploadSessionLookupError('not_closed', None)

not_found = UploadSessionLookupError('not_found', None)

other = UploadSessionLookupError('other', None)

class dropbox.files.**UploadSessionOffsetError**(*correct_offset=None*)
Bases: `object`

Variables ***correct_offset*** – The offset up to which data has been collected.

correct_offset
The offset up to which data has been collected.

Return type long

class dropbox.files.**UploadSessionStartArg**(*close=None*)
Bases: `object`

Variables ***close*** – If true, the current session will be closed, at which point you won't be able to call *dropbox.dropbox.Dropbox.files_upload_session_append_v2()* anymore with the current session.

close
If true, the current session will be closed, at which point you won't be able to call *dropbox.dropbox.Dropbox.files_upload_session_append_v2()* anymore with the current session.

Return type bool

class dropbox.files.**UploadSessionStartResult**(*session_id=None*)
Bases: `object`

Variables ***session_id*** – A unique identifier for the upload session. Pass this to *dropbox.dropbox.Dropbox.files_upload_session_append_v2()* and *dropbox.dropbox.Dropbox.files_upload_session_finish()*.

session_id
A unique identifier for the upload session. Pass this to *dropbox.dropbox.Dropbox.files_upload_session_ap* and *dropbox.dropbox.Dropbox.files_upload_session_finish()*.

Return type str

class dropbox.files.**UploadWriteFailed**(*reason=None*, *upload_session_id=None*)
Bases: `object`

Variables

- ***reason*** – The reason why the file couldn't be saved.
- ***upload_session_id*** – The upload session ID; this may be used to retry the commit.

reason
The reason why the file couldn't be saved.

Return type *WriteError*

upload_session_id
The upload session ID; this may be used to retry the commit.

Return type str

class dropbox.files.**VideoMetadata**(*dimensions=None*, *location=None*, *time_taken=None*, *duration=None*)
Bases: *dropbox.files.MediaMetadata*

Metadata for a video.

Variables ***duration*** – The duration of the video in milliseconds.

duration
The duration of the video in milliseconds.

Return type long

class dropbox.files.**WriteConflictError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***file*** – There's a file in the way.
- ***folder*** – There's a folder in the way.
- ***file_ancestor*** – There's a file at an ancestor path, so we couldn't create the required parent folders.

file = WriteConflictError('file', None)

file_ancestor = WriteConflictError('file_ancestor', None)

folder = WriteConflictError('folder', None)

is_file()
Check if the union tag is `file`.

Return type bool

is_file_ancestor()
Check if the union tag is `file_ancestor`.

Return type bool

is_folder()
Check if the union tag is `folder`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

other = WriteConflictError('other', None)

class `dropbox.files.`**`WriteError`**(*tag*, *value=None*)

Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***conflict*** (`WriteConflictError`) – Couldn't write to the target path because there was something in the way.
- ***no_write_permission*** – The user doesn't have permissions to write to the target location.
- ***insufficient_space*** – The user doesn't have enough available space (bytes) to write more data.
- ***disallowed_name*** – Dropbox will not save the file or folder because of its name.

classmethod **`conflict`**(*val*)

Create an instance of this class set to the `conflict` tag with value `val`.

Parameters **`val`** (`WriteConflictError`) –

Return type *WriteError*

`disallowed_name` = WriteError('disallowed_name', None)

`get_conflict`()

Couldn't write to the target path because there was something in the way.

Only call this if *`is_conflict()`* is true.

Return type *WriteConflictError*

`get_malformed_path`()

Only call this if *`is_malformed_path()`* is true.

Return type Optional[str]

`insufficient_space` = WriteError('insufficient_space', None)

`is_conflict`()

Check if the union tag is `conflict`.

Return type bool

`is_disallowed_name`()

Check if the union tag is `disallowed_name`.

Return type bool

`is_insufficient_space`()

Check if the union tag is `insufficient_space`.

Return type bool

`is_malformed_path`()

Check if the union tag is `malformed_path`.

Return type bool

`is_no_write_permission`()

Check if the union tag is `no_write_permission`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

classmethod malformed_path(*val*)
Create an instance of this class set to the `malformed_path` tag with value `val`.

Parameters **val** (*Optional[str]*) –

Return type *WriteError*

no_write_permission = WriteError('no_write_permission', None)

other = WriteError('other', None)

class dropbox.files.**WriteMode**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Your intent when writing a file to some path. This is used to determine what constitutes a conflict and what the autorename strategy is. In some situations, the conflict behavior is identical: (a) If the target path doesn't contain anything, the file is always written; no conflict. (b) If the target path contains a folder, it's always a conflict. (c) If the target path contains a file with identical contents, nothing gets written; no conflict. The conflict checking differs in the case where there's a file at the target path with contents different from the contents you're trying to write.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***add*** – Never overwrite the existing file. The autorename strategy is to append a number to the file name. For example, "document.txt" might become "document (2).txt".
- ***overwrite*** – Always overwrite the existing file. The autorename strategy is the same as it is for `add`.
- ***update*** (*str*) – Overwrite if the given "rev" matches the existing file's "rev". The autorename strategy is to append the string "conflicted copy" to the file name. For example, "document.txt" might become "document (conflicted copy).txt" or "document (Panda's conflicted copy).txt".

add = WriteMode('add', None)

get_update()
Overwrite if the given "rev" matches the existing file's "rev". The autorename strategy is to append the string "conflicted copy" to the file name. For example, "document.txt" might become "document (conflicted copy).txt" or "document (Panda's conflicted copy).txt".

Only call this if *`is_update()`* is true.

Return type str

is_add()
Check if the union tag is `add`.

Return type bool

is_overwrite()
Check if the union tag is `overwrite`.

Return type bool

is_update()
Check if the union tag is `update`.

Return type bool

overwrite = WriteMode('overwrite', None)

classmethod update(*val*)
Create an instance of this class set to the `update` tag with value `val`.

Parameters **val** (*str*) –

Return type *WriteMode*

2.5 dropbox.sharing – Sharing

This namespace contains endpoints and data types for creating and managing shared links and shared folders.

class dropbox.sharing.**AccessLevel**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

Defines the access levels for collaborators.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***owner*** – The collaborator is the owner of the shared folder. Owners can view and edit the shared folder as well as set the folder's policies using *dropbox.dropbox.Dropbox.sharing_update_folder_policy()*.
- ***editor*** – The collaborator can both view and edit the shared folder.
- ***viewer*** – The collaborator can only view the shared folder.
- ***viewer_no_comment*** – The collaborator can only view the shared folder and does not have any access to comments.

editor = AccessLevel('editor', None)

is_editor()
Check if the union tag is `editor`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_owner()
Check if the union tag is `owner`.

Return type bool

is_viewer()
Check if the union tag is `viewer`.

Return type bool

is_viewer_no_comment()
Check if the union tag is `viewer_no_comment`.

Return type bool

other = AccessLevel('other', None)

owner = AccessLevel('owner', None)

viewer = AccessLevel('viewer', None)

viewer_no_comment = AccessLevel('viewer_no_comment', None)

class dropbox.sharing.**AclUpdatePolicy**(*tag*, *value=None*)

Bases: dropbox.stone_base.Union

Policy governing who can change a shared folder's access control list (ACL). In other words, who can add, remove, or change the privileges of members.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***owner*** – Only the owner can update the ACL.
- ***editors*** – Any editor can update the ACL. This may be further restricted to editors on the same team.

editors = AclUpdatePolicy('editors', None)

is_editors()

Check if the union tag is `editors`.

Return type bool

is_other()

Check if the union tag is `other`.

Return type bool

is_owner()

Check if the union tag is `owner`.

Return type bool

other = AclUpdatePolicy('other', None)

owner = AclUpdatePolicy('owner', None)

class dropbox.sharing.**AddFileMemberArgs**(*file=None*, *members=None*, *custom_message=None*, *quiet=None*, *access_level=None*, *add_message_as_comment=None*)

Bases: object

Arguments for *dropbox.dropbox.Dropbox.sharing_add_file_member()*.

Variables

- ***file*** – File to which to add members.
- ***members*** – Members to add. Note that even an email address is given, this may result in a user being directy added to the membership if that email is the user's main account email.
- ***custom_message*** – Message to send to added members in their invitation.
- ***quiet*** – Whether added members should be notified via device notifications of their invitation.
- ***access_level*** – AccessLevel union object, describing what access level we want to give new members.
- ***add_message_as_comment*** – If the custom message should be added as a comment on the file.

__init__(*file=None*, *members=None*, *custom_message=None*, *quiet=None*, *access_level=None*, *add_message_as_comment=None*)

access_level
AccessLevel union object, describing what access level we want to give new members.

Return type *AccessLevel*

add_message_as_comment
If the custom message should be added as a comment on the file.

Return type bool

custom_message
Message to send to added members in their invitation.

Return type str

file
File to which to add members.

Return type str

members
Members to add. Note that even an email address is given, this may result in a user being directy added to the membership if that email is the user's main account email.

Return type list of [MemberSelector]

quiet
Whether added members should be notified via device notifications of their invitation.

Return type bool

class dropbox.sharing.**AddFileMemberError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

Errors for *dropbox.dropbox.Dropbox.sharing_add_file_member()*.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***rate_limit*** – The user has reached the rate limit for invitations.
- ***invalid_comment*** – The custom message did not pass comment permissions checks.

classmethod access_error(*val*)
Create an instance of this class set to the `access_error` tag with value `val`.

Parameters **val** (SharingFileAccessError) –

Return type *AddFileMemberError*

get_access_error()
Only call this if *is_access_error()* is true.

Return type *SharingFileAccessError*

get_user_error()
Only call this if *is_user_error()* is true.

Return type *SharingUserError*

invalid_comment = AddFileMemberError('invalid_comment', None)

is_access_error()
Check if the union tag is `access_error`.

Return type bool

is_invalid_comment()
Check if the union tag is `invalid_comment`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_rate_limit()
Check if the union tag is `rate_limit`.

Return type bool

is_user_error()
Check if the union tag is `user_error`.

Return type bool

other = AddFileMemberError('other', None)

rate_limit = AddFileMemberError('rate_limit', None)

classmethod user_error(*val*)
Create an instance of this class set to the `user_error` tag with value `val`.

Parameters **val** (`SharingUserError`) –

Return type *AddFileMemberError*

class dropbox.sharing.**AddFolderMemberArg**(*shared_folder_id=None*, *members=None*, *quiet=None*, *custom_message=None*)

Bases: `object`

Variables

- ***shared_folder_id*** – The ID for the shared folder.
- ***members*** – The intended list of members to add. Added members will receive invites to join the shared folder.
- ***quiet*** – Whether added members should be notified via email and device notifications of their invite.
- ***custom_message*** – Optional message to display to added members in their invitation.

__init__(*shared_folder_id=None*, *members=None*, *quiet=None*, *custom_message=None*)

custom_message
Optional message to display to added members in their invitation.

Return type str

members
The intended list of members to add. Added members will receive invites to join the shared folder.

Return type list of [AddMember]

quiet
Whether added members should be notified via email and device notifications of their invite.

Return type bool

shared_folder_id
The ID for the shared folder.

Return type str

class dropbox.sharing.**AddFolderMemberError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables

- ***access_error*** (SharedFolderAccessError) – Unable to access shared folder.
- ***email_unverified*** – The current user's e-mail address is unverified.
- ***bad_member*** (AddMemberSelectorError) – AddFolderMemberArg.members contains a bad invitation recipient.
- ***cant_share_outside_team*** – Your team policy does not allow sharing outside of the team.
- ***too_many_members*** (*long*) – The value is the member limit that was reached.
- ***too_many_pending_invites*** (*long*) – The value is the pending invite limit that was reached.
- ***rate_limit*** – The current user has hit the limit of invites they can send per day. Try again in 24 hours.
- ***too_many_invitees*** – The current user is trying to share with too many people at once.
- ***insufficient_plan*** – The current user's account doesn't support this action. An example of this is when adding a read-only member. This action can only be performed by users that have upgraded to a Pro or Business plan.
- ***team_folder*** – This action cannot be performed on a team shared folder.
- ***no_permission*** – The current user does not have permission to perform this action.

classmethod access_error(*val*)
Create an instance of this class set to the access_error tag with value val.

Parameters **val** (SharedFolderAccessError) –

Return type *AddFolderMemberError*

classmethod bad_member(*val*)
Create an instance of this class set to the bad_member tag with value val.

Parameters **val** (AddMemberSelectorError) –

Return type *AddFolderMemberError*

cant_share_outside_team = AddFolderMemberError('cant_share_outside_team', None)

email_unverified = AddFolderMemberError('email_unverified', None)

get_access_error()
Unable to access shared folder.

Only call this if *is_access_error()* is true.

Return type *SharedFolderAccessError*

get_bad_member()
`AddFolderMemberArg.members` contains a bad invitation recipient.

Only call this if *is_bad_member()* is true.

Return type *AddMemberSelectorError*

get_too_many_members()
The value is the member limit that was reached.

Only call this if *is_too_many_members()* is true.

Return type long

get_too_many_pending_invites()
The value is the pending invite limit that was reached.

Only call this if *is_too_many_pending_invites()* is true.

Return type long

insufficient_plan = AddFolderMemberError('insufficient_plan', None)

is_access_error()
Check if the union tag is `access_error`.

Return type bool

is_bad_member()
Check if the union tag is `bad_member`.

Return type bool

is_cant_share_outside_team()
Check if the union tag is `cant_share_outside_team`.

Return type bool

is_email_unverified()
Check if the union tag is `email_unverified`.

Return type bool

is_insufficient_plan()
Check if the union tag is `insufficient_plan`.

Return type bool

is_no_permission()
Check if the union tag is `no_permission`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_rate_limit()
Check if the union tag is `rate_limit`.

Return type bool

is_team_folder()
Check if the union tag is `team_folder`.

Return type bool

is_too_many_invitees()
Check if the union tag is `too_many_invitees`.

Return type bool

is_too_many_members()
Check if the union tag is `too_many_members`.

Return type bool

is_too_many_pending_invites()
Check if the union tag is `too_many_pending_invites`.

Return type bool

no_permission = AddFolderMemberError('no_permission', None)

other = AddFolderMemberError('other', None)

rate_limit = AddFolderMemberError('rate_limit', None)

team_folder = AddFolderMemberError('team_folder', None)

too_many_invitees = AddFolderMemberError('too_many_invitees', None)

classmethod too_many_members(*val*)
Create an instance of this class set to the `too_many_members` tag with value `val`.

Parameters **val** (*long*) –

Return type *AddFolderMemberError*

classmethod too_many_pending_invites(*val*)
Create an instance of this class set to the `too_many_pending_invites` tag with value `val`.

Parameters **val** (*long*) –

Return type *AddFolderMemberError*

class dropbox.sharing.**AddMember**(*member=None*, *access_level=None*)
Bases: `object`

The member and type of access the member should have when added to a shared folder.

Variables

- ***member*** – The member to add to the shared folder.
- ***access_level*** – The access level to grant `member` to the shared folder. `AccessLevel.owner` is disallowed.

__init__(*member=None*, *access_level=None*)

access_level
The access level to grant `member` to the shared folder. `AccessLevel.owner` is disallowed.

Return type *AccessLevel*

member
The member to add to the shared folder.

Return type *MemberSelector*

class dropbox.sharing.**AddMemberSelectorError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***automatic_group*** – Automatically created groups can only be added to team folders.
- ***invalid_dropbox_id*** (*str*) – The value is the ID that could not be identified.
- ***invalid_email*** (*str*) – The value is the e-mail address that is malformed.
- ***unverified_dropbox_id*** (*str*) – The value is the ID of the Dropbox user with an unverified e-mail address. Invite unverified users by e-mail address instead of by their Dropbox ID.
- ***group_deleted*** – At least one of the specified groups in `AddFolderMemberArg.members` is deleted.
- ***group_not_on_team*** – Sharing to a group that is not on the current user's team.

automatic_group = AddMemberSelectorError('automatic_group', None)

get_invalid_dropbox_id()

The value is the ID that could not be identified.

Only call this if *is_invalid_dropbox_id()* is true.

Return type str

get_invalid_email()

The value is the e-mail address that is malformed.

Only call this if *is_invalid_email()* is true.

Return type str

get_unverified_dropbox_id()

The value is the ID of the Dropbox user with an unverified e-mail address. Invite unverified users by e-mail address instead of by their Dropbox ID.

Only call this if *is_unverified_dropbox_id()* is true.

Return type str

group_deleted = AddMemberSelectorError('group_deleted', None)

group_not_on_team = AddMemberSelectorError('group_not_on_team', None)

classmethod invalid_dropbox_id(*val*)

Create an instance of this class set to the `invalid_dropbox_id` tag with value `val`.

Parameters **val** (*str*) –

Return type *AddMemberSelectorError*

classmethod invalid_email(*val*)

Create an instance of this class set to the `invalid_email` tag with value `val`.

Parameters **val** (*str*) –

Return type *AddMemberSelectorError*

is_automatic_group()

Check if the union tag is `automatic_group`.

Return type bool

is_group_deleted()

Check if the union tag is `group_deleted`.

Return type bool

is_group_not_on_team()
Check if the union tag is `group_not_on_team`.

Return type bool

is_invalid_dropbox_id()
Check if the union tag is `invalid_dropbox_id`.

Return type bool

is_invalid_email()
Check if the union tag is `invalid_email`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_unverified_dropbox_id()
Check if the union tag is `unverified_dropbox_id`.

Return type bool

other = AddMemberSelectorError('other', None)

classmethod unverified_dropbox_id(*val*)
Create an instance of this class set to the `unverified_dropbox_id` tag with value `val`.

Parameters **val** (*str*) –

Return type *AddMemberSelectorError*

class dropbox.sharing.**ChangeFileMemberAccessArgs**(*file=None*, *member=None*, *access_level=None*)

Bases: `object`

Arguments for *dropbox.dropbox.Dropbox.sharing_change_file_member_access()*.

Variables

- ***file*** – File for which we are changing a member's access.
- ***member*** – The member whose access we are changing.
- ***access_level*** – The new access level for the member.

__init__(*file=None*, *member=None*, *access_level=None*)

access_level
The new access level for the member.

Return type *AccessLevel*

file
File for which we are changing a member's access.

Return type str

member
The member whose access we are changing.

Return type *MemberSelector*

class dropbox.sharing.**CollectionLinkMetadata**(*url=None*, *visibility=None*, *expires=None*)
Bases: *dropbox.sharing.LinkMetadata*

Metadata for a collection-based shared link.

__init__(*url=None*, *visibility=None*, *expires=None*)

class dropbox.sharing.**CreateSharedLinkArg**(*path=None*, *short_url=None*, *pending_upload=None*)
Bases: object

Variables

- ***path*** – The path to share.
- ***short_url*** – Whether to return a shortened URL.
- ***pending_upload*** – If it's okay to share a path that does not yet exist, set this to either PendingUploadMode.file or PendingUploadMode.folder to indicate whether to assume it's a file or folder.

__init__(*path=None*, *short_url=None*, *pending_upload=None*)

path
The path to share.

Return type str

pending_upload
If it's okay to share a path that does not yet exist, set this to either PendingUploadMode.file or PendingUploadMode.folder to indicate whether to assume it's a file or folder.

Return type *PendingUploadMode*

short_url
Whether to return a shortened URL.

Return type bool

class dropbox.sharing.**CreateSharedLinkError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

get_path()
Only call this if *is_path()* is true.

Return type files.LookupError_validator

is_other()
Check if the union tag is other.

Return type bool

is_path()
Check if the union tag is path.

Return type bool

other = CreateSharedLinkError('other', None)

classmethod path(*val*)
Create an instance of this class set to the path tag with value val.

Parameters **val** (*files.LookupError_validator*) –

Return type *CreateSharedLinkError*

class `dropbox.sharing.`**`CreateSharedLinkWithSettingsArg`**(*path=None*, *settings=None*)

Bases: `object`

Variables

- ***path*** – The path to be shared by the shared link
- ***settings*** – The requested settings for the newly created shared link

`__init__`(*path=None*, *settings=None*)

`path`

The path to be shared by the shared link

Return type str

`settings`

The requested settings for the newly created shared link

Return type *SharedLinkSettings*

class `dropbox.sharing.`**`CreateSharedLinkWithSettingsError`**(*tag*, *value=None*)

Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***email_not_verified*** – User's email should be verified
- ***shared_link_already_exists*** – The shared link already exists
- ***settings_error*** (`SharedLinkSettingsError`) – There is an error with the given settings
- ***access_denied*** – Access to the requested path is forbidden

`access_denied` = CreateSharedLinkWithSettingsError('access_denied', None)

`email_not_verified` = CreateSharedLinkWithSettingsError('email_not_verified', None)

`get_path`()

Only call this if *`is_path()`* is true.

Return type files.LookupError_validator

`get_settings_error`()

There is an error with the given settings

Only call this if *`is_settings_error()`* is true.

Return type *SharedLinkSettingsError*

`is_access_denied`()

Check if the union tag is `access_denied`.

Return type bool

`is_email_not_verified`()

Check if the union tag is `email_not_verified`.

Return type bool

`is_path`()

Check if the union tag is `path`.

Return type bool

is_settings_error()
Check if the union tag is `settings_error`.

Return type bool

is_shared_link_already_exists()
Check if the union tag is `shared_link_already_exists`.

Return type bool

classmethod path(*val*)
Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (*files.LookupError_validator*) –

Return type *CreateSharedLinkWithSettingsError*

classmethod settings_error(*val*)
Create an instance of this class set to the `settings_error` tag with value `val`.

Parameters **val** (SharedLinkSettingsError) –

Return type *CreateSharedLinkWithSettingsError*

shared_link_already_exists = CreateSharedLinkWithSettingsError('shared_link_already_exists', None)

class dropbox.sharing.**FileAction**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

Sharing actions that may be taken on files.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***edit_contents*** – Change or edit contents of the file.
- ***invite_viewer*** – Add a member with view permissions.
- ***invite_viewer_no_comment*** – Add a member with view permissions but no comment permissions.
- ***unshare*** – Stop sharing this file.
- ***relinquish_membership*** – Relinquish one's own membership to the file.
- ***share_link*** – This action is deprecated. Use create_link instead.
- ***create_link*** – Create a shared link to the file.

create_link = FileAction('create_link', None)

edit_contents = FileAction('edit_contents', None)

invite_viewer = FileAction('invite_viewer', None)

invite_viewer_no_comment = FileAction('invite_viewer_no_comment', None)

is_create_link()
Check if the union tag is `create_link`.

Return type bool

is_edit_contents()
Check if the union tag is `edit_contents`.

Return type bool

is_invite_viewer()
Check if the union tag is `invite_viewer`.

Return type bool

is_invite_viewer_no_comment()
Check if the union tag is `invite_viewer_no_comment`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_relinquish_membership()
Check if the union tag is `relinquish_membership`.

Return type bool

is_share_link()
Check if the union tag is `share_link`.

Return type bool

is_unshare()
Check if the union tag is `unshare`.

Return type bool

other = FileAction('other', None)

relinquish_membership = FileAction('relinquish_membership', None)

share_link = FileAction('share_link', None)

unshare = FileAction('unshare', None)

class dropbox.sharing.**FileErrorResult**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***file_not_found_error*** (*str*) – File specified by id was not found.
- ***invalid_file_action_error*** (*str*) – User does not have permission to take the specified action on the file.
- ***permission_denied_error*** (*str*) – User does not have permission to access file specified by file.Id.

classmethod file_not_found_error(*val*)
Create an instance of this class set to the `file_not_found_error` tag with value `val`.

Parameters **val** (*str*) –

Return type *FileErrorResult*

get_file_not_found_error()
File specified by id was not found.

Only call this if *`is_file_not_found_error()`* is true.

Return type str

get_invalid_file_action_error()

User does not have permission to take the specified action on the file.

Only call this if *is_invalid_file_action_error()* is true.

Return type str

get_permission_denied_error()

User does not have permission to access file specified by file.Id.

Only call this if *is_permission_denied_error()* is true.

Return type str

classmethod invalid_file_action_error(*val*)

Create an instance of this class set to the `invalid_file_action_error` tag with value `val`.

Parameters **val** (*str*) –

Return type *FileErrorResult*

is_file_not_found_error()

Check if the union tag is `file_not_found_error`.

Return type bool

is_invalid_file_action_error()

Check if the union tag is `invalid_file_action_error`.

Return type bool

is_other()

Check if the union tag is `other`.

Return type bool

is_permission_denied_error()

Check if the union tag is `permission_denied_error`.

Return type bool

other = FileErrorResult('other', None)

classmethod permission_denied_error(*val*)

Create an instance of this class set to the `permission_denied_error` tag with value `val`.

Parameters **val** (*str*) –

Return type *FileErrorResult*

class dropbox.sharing.**FileLinkMetadata**(*url=None*, *name=None*, *link_permissions=None*, *client_modified=None*, *server_modified=None*, *rev=None*, *size=None*, *id=None*, *expires=None*, *path_lower=None*, *team_member_info=None*, *content_owner_team_info=None*)

Bases: *dropbox.sharing.SharedLinkMetadata*

The metadata of a file shared link

Variables

- ***client_modified*** – The modification time set by the desktop client when the file was added to Dropbox. Since this time is not verified (the Dropbox server stores whatever the desktop client sends up), this should only be used for display purposes (such as sorting) and not, for example, to determine if a file has changed or not.

- ***server_modified*** – The last time the file was modified on Dropbox.
- ***rev*** – A unique identifier for the current revision of a file. This field is the same rev as elsewhere in the API and can be used to detect changes and avoid conflicts.
- ***size*** – The file size in bytes.

__init__(*url=None*, *name=None*, *link_permissions=None*, *client_modified=None*, *server_modified=None*, *rev=None*, *size=None*, *id=None*, *expires=None*, *path_lower=None*, *team_member_info=None*, *content_owner_team_info=None*)

client_modified

The modification time set by the desktop client when the file was added to Dropbox. Since this time is not verified (the Dropbox server stores whatever the desktop client sends up), this should only be used for display purposes (such as sorting) and not, for example, to determine if a file has changed or not.

Return type datetime.datetime

rev

A unique identifier for the current revision of a file. This field is the same rev as elsewhere in the API and can be used to detect changes and avoid conflicts.

Return type str

server_modified

The last time the file was modified on Dropbox.

Return type datetime.datetime

size

The file size in bytes.

Return type long

class dropbox.sharing.**FileMemberActionError**(*tag*, *value=None*)

Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***invalid_member*** – Specified member was not found.
- ***no_permission*** – User does not have permission to perform this action on this member.
- ***access_error*** (`SharingFileAccessError`) – Specified file was invalid or user does not have access.

classmethod access_error(*val*)

Create an instance of this class set to the `access_error` tag with value `val`.

Parameters **val** (`SharingFileAccessError`) –

Return type *FileMemberActionError*

get_access_error()

Specified file was invalid or user does not have access.

Only call this if *is_access_error()* is true.

Return type *SharingFileAccessError*

invalid_member = FileMemberActionError('invalid_member', None)

is_access_error()

Check if the union tag is `access_error`.

Return type bool

is_invalid_member()
Check if the union tag is `invalid_member`.

Return type bool

is_no_permission()
Check if the union tag is `no_permission`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

no_permission = FileMemberActionError('no_permission', None)

other = FileMemberActionError('other', None)

class dropbox.sharing.**FileMemberActionIndividualResult**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***success*** (*Optional[AccessLevel]*) – Member was successfully removed from this file. If AccessLevel is given, the member still has access via a parent shared folder.
- ***member_error*** (FileMemberActionError) – User was not able to perform this action.

get_member_error()
User was not able to perform this action.

Only call this if *is_member_error()* is true.

Return type *FileMemberActionError*

get_success()
Member was successfully removed from this file. If AccessLevel is given, the member still has access via a parent shared folder.

Only call this if *is_success()* is true.

Return type *AccessLevel*

is_member_error()
Check if the union tag is `member_error`.

Return type bool

is_success()
Check if the union tag is `success`.

Return type bool

classmethod member_error(*val*)
Create an instance of this class set to the `member_error` tag with value `val`.

Parameters **val** (FileMemberActionError) –

Return type *FileMemberActionIndividualResult*

classmethod success(*val*)
Create an instance of this class set to the `success` tag with value `val`.

Parameters **val** (`AccessLevel`) –

Return type *FileMemberActionIndividualResult*

class `dropbox.sharing.`**FileMemberActionResult**(*member=None*, *result=None*)
Bases: `object`

Per-member result for *dropbox.dropbox.Dropbox.sharing_remove_file_member_2()* or *dropbox.dropbox.Dropbox.sharing_add_file_member()* or *dropbox.dropbox.Dropbox.sharing_change_file_member_access()*.

Variables

- ***member*** – One of specified input members.
- ***result*** – The outcome of the action on this member.

__init__(*member=None*, *result=None*)

member
One of specified input members.

Return type *MemberSelector*

result
The outcome of the action on this member.

Return type *FileMemberActionIndividualResult*

class `dropbox.sharing.`**FileMemberRemoveActionResult**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***success*** (`MemberAccessLevelResult`) – Member was successfully removed from this file.
- ***member_error*** (`FileMemberActionError`) – User was not able to remove this member.

get_member_error()
User was not able to remove this member.

Only call this if *is_member_error()* is true.

Return type *FileMemberActionError*

get_success()
Member was successfully removed from this file.

Only call this if *is_success()* is true.

Return type *MemberAccessLevelResult*

is_member_error()
Check if the union tag is `member_error`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_success()
Check if the union tag is `success`.

Return type bool

classmethod member_error(*val*)
Create an instance of this class set to the `member_error` tag with value `val`.

Parameters **val** (`FileMemberActionError`) –

Return type *FileMemberRemoveActionResult*

other = FileMemberRemoveActionResult('other', None)

classmethod success(*val*)
Create an instance of this class set to the `success` tag with value `val`.

Parameters **val** (`MemberAccessLevelResult`) –

Return type *FileMemberRemoveActionResult*

class dropbox.sharing.**FilePermission**(*action=None*, *allow=None*, *reason=None*)
Bases: `object`

Whether the user is allowed to take the sharing action on the file.

Variables

- ***action*** – The action that the user may wish to take on the file.
- ***allow*** – True if the user is allowed to take the action.
- ***reason*** – The reason why the user is denied the permission. Not present if the action is allowed

__init__(*action=None*, *allow=None*, *reason=None*)

action
The action that the user may wish to take on the file.

Return type *FileAction*

allow
True if the user is allowed to take the action.

Return type bool

reason
The reason why the user is denied the permission. Not present if the action is allowed

Return type *PermissionDeniedReason*

class dropbox.sharing.**FolderAction**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Actions that may be taken on shared folders.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***change_options*** – Change folder options, such as who can be invited to join the folder.
- ***edit_contents*** – Change or edit contents of the folder.

- *invite_editor* – Invite a user or group to join the folder with read and write permission.
- *invite_viewer* – Invite a user or group to join the folder with read permission.
- *invite_viewer_no_comment* – Invite a user or group to join the folder with read permission but no comment permissions.
- *relinquish_membership* – Relinquish one's own membership in the folder.
- *unmount* – Unmount the folder.
- *unshare* – Stop sharing this folder.
- *leave_a_copy* – Keep a copy of the contents upon leaving or being kicked from the folder.
- *share_link* – This action is deprecated. Use create_link instead.
- *create_link* – Create a shared link for folder.

change_options = FolderAction('change_options', None)

create_link = FolderAction('create_link', None)

edit_contents = FolderAction('edit_contents', None)

invite_editor = FolderAction('invite_editor', None)

invite_viewer = FolderAction('invite_viewer', None)

invite_viewer_no_comment = FolderAction('invite_viewer_no_comment', None)

is_change_options()
Check if the union tag is `change_options`.

Return type bool

is_create_link()
Check if the union tag is `create_link`.

Return type bool

is_edit_contents()
Check if the union tag is `edit_contents`.

Return type bool

is_invite_editor()
Check if the union tag is `invite_editor`.

Return type bool

is_invite_viewer()
Check if the union tag is `invite_viewer`.

Return type bool

is_invite_viewer_no_comment()
Check if the union tag is `invite_viewer_no_comment`.

Return type bool

is_leave_a_copy()
Check if the union tag is `leave_a_copy`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_relinquish_membership()
Check if the union tag is `relinquish_membership`.

Return type bool

is_share_link()
Check if the union tag is `share_link`.

Return type bool

is_unmount()
Check if the union tag is `unmount`.

Return type bool

is_unshare()
Check if the union tag is `unshare`.

Return type bool

leave_a_copy = FolderAction('leave_a_copy', None)

other = FolderAction('other', None)

relinquish_membership = FolderAction('relinquish_membership', None)

share_link = FolderAction('share_link', None)

unmount = FolderAction('unmount', None)

unshare = FolderAction('unshare', None)

class dropbox.sharing.**FolderLinkMetadata**(*url=None*, *name=None*, *link_permissions=None*, *id=None*, *expires=None*, *path_lower=None*, *team_member_info=None*, *content_owner_team_info=None*)
Bases: *dropbox.sharing.SharedLinkMetadata*

The metadata of a folder shared link

__init__(*url=None*, *name=None*, *link_permissions=None*, *id=None*, *expires=None*, *path_lower=None*, *team_member_info=None*, *content_owner_team_info=None*)

class dropbox.sharing.**FolderPermission**(*action=None*, *allow=None*, *reason=None*)
Bases: `object`

Whether the user is allowed to take the action on the shared folder.

Variables

- ***action*** – The action that the user may wish to take on the folder.
- ***allow*** – True if the user is allowed to take the action.
- ***reason*** – The reason why the user is denied the permission. Not present if the action is allowed, or if no reason is available.

__init__(*action=None*, *allow=None*, *reason=None*)

action
The action that the user may wish to take on the folder.

Return type *FolderAction*

allow
True if the user is allowed to take the action.

Return type bool

reason
The reason why the user is denied the permission. Not present if the action is allowed, or if no reason is available.

Return type *PermissionDeniedReason*

class `dropbox.sharing.`**`FolderPolicy`**(*acl_update_policy=None*, *shared_link_policy=None*, *member_policy=None*, *resolved_member_policy=None*)

Bases: `object`

A set of policies governing membership and privileges for a shared folder.

Variables

- ***member_policy*** – Who can be a member of this shared folder, as set on the folder itself. The effective policy may differ from this value if the team-wide policy is more restrictive. Present only if the folder is owned by a team.
- ***resolved_member_policy*** – Who can be a member of this shared folder, taking into account both the folder and the team-wide policy. This value may differ from that of member_policy if the team-wide policy is more restrictive than the folder policy. Present only if the folder is owned by a team.
- ***acl_update_policy*** – Who can add and remove members from this shared folder.
- ***shared_link_policy*** – Who links can be shared with.

__init__(*acl_update_policy=None*, *shared_link_policy=None*, *member_policy=None*, *resolved_member_policy=None*)

acl_update_policy
Who can add and remove members from this shared folder.

Return type *AclUpdatePolicy*

member_policy
Who can be a member of this shared folder, as set on the folder itself. The effective policy may differ from this value if the team-wide policy is more restrictive. Present only if the folder is owned by a team.

Return type *MemberPolicy*

resolved_member_policy
Who can be a member of this shared folder, taking into account both the folder and the team-wide policy. This value may differ from that of member_policy if the team-wide policy is more restrictive than the folder policy. Present only if the folder is owned by a team.

Return type *MemberPolicy*

shared_link_policy
Who links can be shared with.

Return type *SharedLinkPolicy*

class `dropbox.sharing.`**`GetFileMetadataArg`**(*file=None*, *actions=None*)

Bases: `object`

Arguments of `dropbox.dropbox.Dropbox.sharing_get_file_metadata()`

Variables

- ***file*** – The file to query.

- ***actions*** – File actions to query.

__init__ (*file=None*, *actions=None*)

actions
File actions to query.

Return type list of [FileAction]

file
The file to query.

Return type str

class `dropbox.sharing.`**GetFileMetadataBatchArg** (*files=None*, *actions=None*)
Bases: `object`

Arguments of *dropbox.dropbox.Dropbox.sharing_get_file_metadata_batch()*

Variables

- ***files*** – The files to query.
- ***actions*** – File actions to query.

__init__ (*files=None*, *actions=None*)

actions
File actions to query.

Return type list of [FileAction]

files
The files to query.

Return type list of [str]

class `dropbox.sharing.`**GetFileMetadataBatchResult** (*file=None*, *result=None*)
Bases: `object`

Per file results of *dropbox.dropbox.Dropbox.sharing_get_file_metadata_batch()*

Variables

- ***file*** – This is the input file identifier corresponding to one of `GetFileMetadataBatchArg.files`.
- ***result*** – The result for this particular file

__init__ (*file=None*, *result=None*)

file
This is the input file identifier corresponding to one of `GetFileMetadataBatchArg.files`.

Return type str

result
The result for this particular file

Return type *GetFileMetadataIndividualResult*

class `dropbox.sharing.`**GetFileMetadataError** (*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Error result for *dropbox.dropbox.Dropbox.sharing_get_file_metadata()*.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

classmethod access_error(*val*)
Create an instance of this class set to the `access_error` tag with value `val`.

Parameters **val** (`SharingFileAccessError`) –

Return type *GetFileMetadataError*

get_access_error()
Only call this if *`is_access_error()`* is true.

Return type *SharingFileAccessError*

get_user_error()
Only call this if *`is_user_error()`* is true.

Return type *SharingUserError*

is_access_error()
Check if the union tag is `access_error`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_user_error()
Check if the union tag is `user_error`.

Return type bool

other = GetFileMetadataError('other', None)

classmethod user_error(*val*)
Create an instance of this class set to the `user_error` tag with value `val`.

Parameters **val** (`SharingUserError`) –

Return type *GetFileMetadataError*

class dropbox.sharing.**GetFileMetadataIndividualResult**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***metadata*** (`SharedFileMetadata`) – The result for this file if it was successful.
- ***access_error*** (`SharingFileAccessError`) – The result for this file if it was an error.

classmethod access_error(*val*)
Create an instance of this class set to the `access_error` tag with value `val`.

Parameters **val** (`SharingFileAccessError`) –

Return type *GetFileMetadataIndividualResult*

get_access_error()
The result for this file if it was an error.

Only call this if *`is_access_error()`* is true.

Return type *SharingFileAccessError*

get_metadata()
The result for this file if it was successful.

Only call this if *is_metadata()* is true.

Return type *SharedFileMetadata*

is_access_error()
Check if the union tag is `access_error`.

Return type bool

is_metadata()
Check if the union tag is `metadata`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

classmethod metadata(*val*)
Create an instance of this class set to the `metadata` tag with value `val`.

Parameters **val** (`SharedFileMetadata`) –

Return type *GetFileMetadataIndividualResult*

other = GetFileMetadataIndividualResult('other', None)

class dropbox.sharing.**GetMetadataArgs**(*shared_folder_id=None*, *actions=None*)
Bases: `object`

Variables

- ***shared_folder_id*** – The ID for the shared folder.
- ***actions*** – This is a list indicating whether the returned folder data will include a boolean value `FolderPermission.allow` that describes whether the current user can perform the FolderAction on the folder.

__init__(*shared_folder_id=None*, *actions=None*)

actions
This is a list indicating whether the returned folder data will include a boolean value `FolderPermission.allow` that describes whether the current user can perform the FolderAction on the folder.

Return type list of [FolderAction]

shared_folder_id
The ID for the shared folder.

Return type str

dropbox.sharing.**GetSharedLinkFileArg**
alias of *GetSharedLinkMetadataArg*

class dropbox.sharing.**GetSharedLinkFileError**(*tag*, *value=None*)
Bases: *dropbox.sharing.SharedLinkError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***shared_link_is_directory*** – Directories cannot be retrieved by this endpoint.

is_shared_link_is_directory()
Check if the union tag is `shared_link_is_directory`.

Return type bool

shared_link_is_directory = GetSharedLinkFileError('shared_link_is_directory', None)

class dropbox.sharing.**GetSharedLinkMetadataArg**(*url=None*, *path=None*, *link_password=None*)

Bases: `object`

Variables

- ***url*** – URL of the shared link.
- ***path*** – If the shared link is to a folder, this parameter can be used to retrieve the metadata for a specific file or sub-folder in this folder. A relative path should be used.
- ***link_password*** – If the shared link has a password, this parameter can be used.

__init__(*url=None*, *path=None*, *link_password=None*)

link_password
If the shared link has a password, this parameter can be used.

Return type str

path
If the shared link is to a folder, this parameter can be used to retrieve the metadata for a specific file or sub-folder in this folder. A relative path should be used.

Return type str

url
URL of the shared link.

Return type str

class dropbox.sharing.**GetSharedLinksArg**(*path=None*)
Bases: `object`

Variables ***path*** – See *dropbox.dropbox.Dropbox.sharing_get_shared_links()* description.

__init__(*path=None*)

path
See *dropbox.dropbox.Dropbox.sharing_get_shared_links()* description.

Return type str

class dropbox.sharing.**GetSharedLinksError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

get_path()
Only call this if *is_path()* is true.

Return type Optional[str]

is_other()
Check if the union tag is `other`.

Return type bool

is_path()
Check if the union tag is `path`.

Return type bool

other = GetSharedLinksError('other', None)

classmethod path(*val*)
Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (*Optional[str]*) –

Return type *GetSharedLinksError*

class dropbox.sharing.**GetSharedLinksResult**(*links=None*)
Bases: `object`

Variables ***links*** – Shared links applicable to the path argument.

__init__(*links=None*)

links
Shared links applicable to the path argument.

Return type list of [LinkMetadata]

class dropbox.sharing.**GroupInfo**(*group_name=None*, *group_id=None*, *group_management_type=None*, *group_type=None*, *is_owner=None*, *same_team=None*, *group_external_id=None*, *member_count=None*)
Bases: `dropbox.team_common.GroupSummary`

The information about a group. Groups is a way to manage a list of users who need same access permission to the shared folder.

Variables

- ***group_type*** – The type of group.
- ***is_owner*** – If the current user is an owner of the group.
- ***same_team*** – If the group is owned by the current user's team.

__init__(*group_name=None*, *group_id=None*, *group_management_type=None*, *group_type=None*, *is_owner=None*, *same_team=None*, *group_external_id=None*, *member_count=None*)

group_type
The type of group.

Return type team_common.GroupType_validator

is_owner
If the current user is an owner of the group.

Return type bool

same_team
If the group is owned by the current user's team.

Return type bool

class dropbox.sharing.**GroupMembershipInfo**(*access_type=None*, *group=None*, *permissions=None*, *initials=None*, *is_inherited=None*)
Bases: *dropbox.sharing.MembershipInfo*

The information about a group member of the shared content.

Variables ***group*** – The information about the membership group.

__init__(*access_type=None*, *group=None*, *permissions=None*, *initials=None*, *is_inherited=None*)

group
The information about the membership group.

Return type *GroupInfo*

class dropbox.sharing.**InsufficientQuotaAmounts**(*space_needed=None*, *space_shortage=None*, *space_left=None*)
Bases: object

Variables

- ***space_needed*** – The amount of space needed to add the item (the size of the item).
- ***space_shortage*** – The amount of extra space needed to add the item.
- ***space_left*** – The amount of space left in the user's Dropbox, less than space_needed.

__init__(*space_needed=None*, *space_shortage=None*, *space_left=None*)

space_left
The amount of space left in the user's Dropbox, less than space_needed.

Return type long

space_needed
The amount of space needed to add the item (the size of the item).

Return type long

space_shortage
The amount of extra space needed to add the item.

Return type long

class dropbox.sharing.**InviteeInfo**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

Information about the recipient of a shared content invitation.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***email*** (*str*) – E-mail address of invited user.

classmethod email(*val*)
Create an instance of this class set to the `email` tag with value `val`.

Parameters **val** (*str*) –

Return type *InviteeInfo*

get_email()
E-mail address of invited user.

Only call this if *is_email()* is true.

Return type str

is_email()
Check if the union tag is `email`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

other = InviteeInfo('other', None)

class dropbox.sharing.**InviteeMembershipInfo**(*access_type=None*, *invitee=None*, *permissions=None*, *initials=None*, *is_inherited=None*, *user=None*)

Bases: *dropbox.sharing.MembershipInfo*

Information about an invited member of a shared content.

Variables

- ***invitee*** – Recipient of the invitation.
- ***user*** – The user this invitation is tied to, if available.

__init__(*access_type=None*, *invitee=None*, *permissions=None*, *initials=None*, *is_inherited=None*, *user=None*)

invitee

Recipient of the invitation.

Return type *InviteeInfo*

user

The user this invitation is tied to, if available.

Return type *UserInfo*

class dropbox.sharing.**JobError**(*tag*, *value=None*)

Bases: dropbox.stone_base.Union

Error occurred while performing an asynchronous job from *dropbox.dropbox.Dropbox.sharing_unshare_folder(* or *dropbox.dropbox.Dropbox.sharing_remove_folder_member()*.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***unshare_folder_error*** (UnshareFolderError) – Error occurred while performing *dropbox.dropbox.Dropbox.sharing_unshare_folder()* action.
- ***remove_folder_member_error*** (RemoveFolderMemberError) – Error occurred while performing *dropbox.dropbox.Dropbox.sharing_remove_folder_member()* action.
- ***relinquish_folder_membership_error*** (RelinquishFolderMembershipError) – Error occurred while performing *dropbox.dropbox.Dropbox.sharing_relinquish_folder_memb* action.

get_relinquish_folder_membership_error()

Error occurred while performing *dropbox.dropbox.Dropbox.sharing_relinquish_folder_membership(* action.

Only call this if *is_relinquish_folder_membership_error()* is true.

Return type *RelinquishFolderMembershipError*

get_remove_folder_member_error()

Error occurred while performing *dropbox.dropbox.Dropbox.sharing_remove_folder_member()* action.

Only call this if *is_remove_folder_member_error()* is true.

Return type *RemoveFolderMemberError*

get_unshare_folder_error()

Error occurred while performing `dropbox.dropbox.Dropbox.sharing_unshare_folder()` action.

Only call this if `is_unshare_folder_error()` is true.

Return type *UnshareFolderError*

is_other()

Check if the union tag is `other`.

Return type bool

is_relinquish_folder_membership_error()

Check if the union tag is `relinquish_folder_membership_error`.

Return type bool

is_remove_folder_member_error()

Check if the union tag is `remove_folder_member_error`.

Return type bool

is_unshare_folder_error()

Check if the union tag is `unshare_folder_error`.

Return type bool

other = JobError('other', None)

classmethod relinquish_folder_membership_error(*val*)

Create an instance of this class set to the `relinquish_folder_membership_error` tag with value `val`.

Parameters **val** (`RelinquishFolderMembershipError`) –

Return type *JobError*

classmethod remove_folder_member_error(*val*)

Create an instance of this class set to the `remove_folder_member_error` tag with value `val`.

Parameters **val** (`RemoveFolderMemberError`) –

Return type *JobError*

classmethod unshare_folder_error(*val*)

Create an instance of this class set to the `unshare_folder_error` tag with value `val`.

Parameters **val** (`UnshareFolderError`) –

Return type *JobError*

class dropbox.sharing.**JobStatus**(*tag*, *value=None*)

Bases: `dropbox.async.PollResultBase`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***complete*** – The asynchronous job has finished.
- ***failed*** (`JobError`) – The asynchronous job returned an error.

complete = JobStatus('complete', None)

classmethod failed(*val*)
Create an instance of this class set to the `failed` tag with value `val`.

Parameters **val** (`JobError`) –

Return type *JobStatus*

get_failed()
The asynchronous job returned an error.

Only call this if *`is_failed()`* is true.

Return type *JobError*

is_complete()
Check if the union tag is `complete`.

Return type bool

is_failed()
Check if the union tag is `failed`.

Return type bool

class dropbox.sharing.**LinkMetadata**(*url=None*, *visibility=None*, *expires=None*)
Bases: `object`

Metadata for a shared link. This can be either a *`PathLinkMetadata`* or *`CollectionLinkMetadata`*.

Variables

- ***url*** – URL of the shared link.
- ***visibility*** – Who can access the link.
- ***expires*** – Expiration time, if set. By default the link won't expire.

__init__(*url=None*, *visibility=None*, *expires=None*)

expires
Expiration time, if set. By default the link won't expire.

Return type datetime.datetime

url
URL of the shared link.

Return type str

visibility
Who can access the link.

Return type *Visibility*

class dropbox.sharing.**LinkPermissions**(*can_revoke=None*, *resolved_visibility=None*, *requested_visibility=None*, *revoke_failure_reason=None*)
Bases: `object`

Variables

- ***resolved_visibility*** – The current visibility of the link after considering the shared links policies of the the team (in case the link's owner is part of a team) and the shared folder (in case the linked file is part of a shared folder). This field is shown only if the caller has access to this info (the link's owner always has access to this data).

- ***requested_visibility*** – The shared link's requested visibility. This can be overridden by the team and shared folder policies. The final visibility, after considering these policies, can be found in `resolved_visibility`. This is shown only if the caller is the link's owner.
- ***can_revoke*** – Whether the caller can revoke the shared link
- ***revoke_failure_reason*** – The failure reason for revoking the link. This field will only be present if the `can_revoke` is `False`.

__init__(*can_revoke=None*, *resolved_visibility=None*, *requested_visibility=None*, *revoke_failure_reason=None*)

can_revoke

Whether the caller can revoke the shared link

Return type bool

requested_visibility

The shared link's requested visibility. This can be overridden by the team and shared folder policies. The final visibility, after considering these policies, can be found in `resolved_visibility`. This is shown only if the caller is the link's owner.

Return type *RequestedVisibility*

resolved_visibility

The current visibility of the link after considering the shared links policies of the the team (in case the link's owner is part of a team) and the shared folder (in case the linked file is part of a shared folder). This field is shown only if the caller has access to this info (the link's owner always has access to this data).

Return type *ResolvedVisibility*

revoke_failure_reason

The failure reason for revoking the link. This field will only be present if the `can_revoke` is `False`.

Return type *SharedLinkAccessFailureReason*

class `dropbox.sharing.`**ListFileMembersArg**(*file=None*, *actions=None*, *include_inherited=None*, *limit=None*)

Bases: `object`

Arguments for *`dropbox.dropbox.Dropbox.sharing_list_file_members()`*.

Variables

- ***file*** – The file for which you want to see members.
- ***actions*** – The actions for which to return permissions on a member
- ***include_inherited*** – Whether to include members who only have access from a parent shared folder.
- ***limit*** – Number of members to return max per query. Defaults to 100 if no limit is specified.

__init__(*file=None*, *actions=None*, *include_inherited=None*, *limit=None*)

actions

The actions for which to return permissions on a member

Return type list of [MemberAction]

file

The file for which you want to see members.

Return type str

include_inherited
Whether to include members who only have access from a parent shared folder.

Return type bool

limit
Number of members to return max per query. Defaults to 100 if no limit is specified.

Return type long

class dropbox.sharing.**ListFileMembersBatchArg**(*files=None*, *limit=None*)
Bases: object

Arguments for *dropbox.dropbox.Dropbox.sharing_list_file_members_batch()*.

Variables

- ***files*** – Files for which to return members.
- ***limit*** – Number of members to return max per query. Defaults to 10 if no limit is specified.

__init__(*files=None*, *limit=None*)

files
Files for which to return members.

Return type list of [str]

limit
Number of members to return max per query. Defaults to 10 if no limit is specified.

Return type long

class dropbox.sharing.**ListFileMembersBatchResult**(*file=None*, *result=None*)
Bases: object

Per-file result for *dropbox.dropbox.Dropbox.sharing_list_file_members_batch()*.

Variables

- ***file*** – This is the input file identifier, whether an ID or a path.
- ***result*** – The result for this particular file

__init__(*file=None*, *result=None*)

file
This is the input file identifier, whether an ID or a path.

Return type str

result
The result for this particular file

Return type *ListFileMembersIndividualResult*

class dropbox.sharing.**ListFileMembersContinueArg**(*cursor=None*)
Bases: object

Arguments for *dropbox.dropbox.Dropbox.sharing_list_file_members_continue()*.

Variables ***cursor*** – The cursor returned by your last call to *dropbox.dropbox.Dropbox.sharing_list_file_members()*, *dropbox.dropbox.Dropbox.sharing_list_file_members_continue()*, or *dropbox.dropbox.Dropbox.sharing_list_file_members_batch()*.

__init__(*cursor=None*)

cursor
The cursor returned by your last call to *dropbox.dropbox.Dropbox.sharing_list_file_members()*, *dropbox.dropbox.Dropbox.sharing_list_file_members_continue()*, or *dropbox.dropbox.Dropbox.sharing_list_file_members_batch()*.

Return type str

class dropbox.sharing.**ListFileMembersContinueError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

Error for *dropbox.dropbox.Dropbox.sharing_list_file_members_continue()*.

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables ***invalid_cursor*** – ListFileMembersContinueArg.cursor is invalid.

classmethod access_error(*val*)
Create an instance of this class set to the access_error tag with value val.

Parameters **val** (SharingFileAccessError) –

Return type *ListFileMembersContinueError*

get_access_error()
Only call this if *is_access_error()* is true.

Return type *SharingFileAccessError*

get_user_error()
Only call this if *is_user_error()* is true.

Return type *SharingUserError*

invalid_cursor = ListFileMembersContinueError('invalid_cursor', None)

is_access_error()
Check if the union tag is access_error.

Return type bool

is_invalid_cursor()
Check if the union tag is invalid_cursor.

Return type bool

is_other()
Check if the union tag is other.

Return type bool

is_user_error()
Check if the union tag is user_error.

Return type bool

other = ListFileMembersContinueError('other', None)

classmethod user_error(*val*)
Create an instance of this class set to the user_error tag with value val.

Parameters **val** (SharingUserError) –

Return type *ListFileMembersContinueError*

class dropbox.sharing.**ListFileMembersCountResult**(*members=None*, *member_count=None*)
Bases: object

Variables

- ***members*** – A list of members on this file.
- ***member_count*** – The number of members on this file. This does not include inherited members

__init__(*members=None*, *member_count=None*)

member_count

The number of members on this file. This does not include inherited members

Return type long

members

A list of members on this file.

Return type *SharedFileMembers*

class dropbox.sharing.**ListFileMembersError**(*tag*, *value=None*)

Bases: dropbox.stone_base.Union

Error for *dropbox.dropbox.Dropbox.sharing_list_file_members()*.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

classmethod access_error(*val*)

Create an instance of this class set to the `access_error` tag with value `val`.

Parameters **val** (`SharingFileAccessError`) –

Return type *ListFileMembersError*

get_access_error()

Only call this if *is_access_error()* is true.

Return type *SharingFileAccessError*

get_user_error()

Only call this if *is_user_error()* is true.

Return type *SharingUserError*

is_access_error()

Check if the union tag is `access_error`.

Return type bool

is_other()

Check if the union tag is `other`.

Return type bool

is_user_error()

Check if the union tag is `user_error`.

Return type bool

other = ListFileMembersError('other', None)

classmethod user_error(*val*)

Create an instance of this class set to the `user_error` tag with value `val`.

Parameters **val** (`SharingUserError`) –

Return type *ListFileMembersError*

class dropbox.sharing.**ListFileMembersIndividualResult**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables

- ***result*** (ListFileMembersCountResult) – The results of the query for this file if it was successful
- ***access_error*** (SharingFileAccessError) – The result of the query for this file if it was an error.

classmethod access_error(*val*)
Create an instance of this class set to the access_error tag with value val.

Parameters **val** (SharingFileAccessError) –

Return type *ListFileMembersIndividualResult*

get_access_error()
The result of the query for this file if it was an error.

Only call this if *is_access_error()* is true.

Return type *SharingFileAccessError*

get_result()
The results of the query for this file if it was successful

Only call this if *is_result()* is true.

Return type *ListFileMembersCountResult*

is_access_error()
Check if the union tag is access_error.

Return type bool

is_other()
Check if the union tag is other.

Return type bool

is_result()
Check if the union tag is result.

Return type bool

other = ListFileMembersIndividualResult('other', None)

classmethod result(*val*)
Create an instance of this class set to the result tag with value val.

Parameters **val** (ListFileMembersCountResult) –

Return type *ListFileMembersIndividualResult*

class dropbox.sharing.**ListFilesArg**(*limit=None*, *actions=None*)
Bases: object

Arguments for *dropbox.dropbox.Dropbox.sharing_list_received_files()*.

Variables

- ***limit*** – Number of files to return max per query. Defaults to 100 if no limit is specified.

- ***actions*** – File actions to query.

__init__(*limit=None*, *actions=None*)

actions
File actions to query.

Return type list of [FileAction]

limit
Number of files to return max per query. Defaults to 100 if no limit is specified.

Return type long

class dropbox.sharing.**ListFilesContinueArg**(*cursor=None*)
Bases: object

Arguments for *dropbox.dropbox.Dropbox.sharing_list_received_files_continue()*.

Variables ***cursor*** – Cursor in ListFilesResult.cursor

__init__(*cursor=None*)

cursor
Cursor in ListFilesResult.cursor

Return type str

class dropbox.sharing.**ListFilesContinueError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

Error results for *dropbox.dropbox.Dropbox.sharing_list_received_files_continue()*.

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables

- ***user_error*** (SharingUserError) – User account had a problem.
- ***invalid_cursor*** – ListFilesContinueArg.cursor is invalid.

get_user_error()
User account had a problem.

Only call this if *is_user_error()* is true.

Return type *SharingUserError*

invalid_cursor = ListFilesContinueError('invalid_cursor', None)

is_invalid_cursor()
Check if the union tag is invalid_cursor.

Return type bool

is_other()
Check if the union tag is other.

Return type bool

is_user_error()
Check if the union tag is user_error.

Return type bool

other = ListFilesContinueError('other', None)

classmethod user_error (*val*)
Create an instance of this class set to the `user_error` tag with value `val`.

Parameters **val** (`SharingUserError`) –

Return type *ListFilesContinueError*

class `dropbox.sharing.`**ListFilesResult** (*entries=None*, *cursor=None*)
Bases: `object`

Success results for *`dropbox.dropbox.Dropbox.sharing_list_received_files()`*.

Variables

- ***entries*** – Information about the files shared with current user.
- ***cursor*** – Cursor used to obtain additional shared files.

__init__ (*entries=None*, *cursor=None*)

cursor
Cursor used to obtain additional shared files.

Return type str

entries
Information about the files shared with current user.

Return type list of [SharedFileMetadata]

class `dropbox.sharing.`**ListFolderMembersArgs** (*shared_folder_id=None*, *actions=None*, *limit=None*)
Bases: *`dropbox.sharing.ListFolderMembersCursorArg`*

Variables ***shared_folder_id*** – The ID for the shared folder.

__init__ (*shared_folder_id=None*, *actions=None*, *limit=None*)

shared_folder_id
The ID for the shared folder.

Return type str

class `dropbox.sharing.`**ListFolderMembersContinueArg** (*cursor=None*)
Bases: `object`

Variables ***cursor*** – The cursor returned by your last call to *`dropbox.dropbox.Dropbox.sharing_list_folder_members()`* or *`dropbox.dropbox.Dropbox.sharing_list_folder_members_continue()`*.

__init__ (*cursor=None*)

cursor
The cursor returned by your last call to *`dropbox.dropbox.Dropbox.sharing_list_folder_members()`* or *`dropbox.dropbox.Dropbox.sharing_list_folder_members_continue()`*.

Return type str

class `dropbox.sharing.`**ListFolderMembersContinueError** (*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***invalid_cursor*** – `ListFolderMembersContinueArg.cursor` is invalid.

classmethod access_error(*val*)
Create an instance of this class set to the `access_error` tag with value `val`.

Parameters **val** (`SharedFolderAccessError`) –

Return type *ListFolderMembersContinueError*

get_access_error()
Only call this if *`is_access_error()`* is true.

Return type *SharedFolderAccessError*

invalid_cursor = ListFolderMembersContinueError('invalid_cursor', None)

is_access_error()
Check if the union tag is `access_error`.

Return type bool

is_invalid_cursor()
Check if the union tag is `invalid_cursor`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

other = ListFolderMembersContinueError('other', None)

class dropbox.sharing.**ListFolderMembersCursorArg**(*actions=None*, *limit=None*)
Bases: `object`

Variables

- ***actions*** – This is a list indicating whether each returned member will include a boolean value `MemberPermission.allow` that describes whether the current user can perform the MemberAction on the member.
- ***limit*** – The maximum number of results that include members, groups and invitees to return per request.

__init__(*actions=None*, *limit=None*)

actions
This is a list indicating whether each returned member will include a boolean value `MemberPermission.allow` that describes whether the current user can perform the MemberAction on the member.

Return type list of [MemberAction]

limit
The maximum number of results that include members, groups and invitees to return per request.

Return type long

class dropbox.sharing.**ListFoldersArgs**(*limit=None*, *actions=None*)
Bases: `object`

Variables

- ***limit*** – The maximum number of results to return per request.

- ***actions*** – This is a list indicating whether each returned folder data entry will include a boolean field `FolderPermission.allow` that describes whether the current user can perform the *FolderAction* on the folder.

__init__(*limit=None*, *actions=None*)

actions

This is a list indicating whether each returned folder data entry will include a boolean field `FolderPermission.allow` that describes whether the current user can perform the *FolderAction* on the folder.

Return type list of [FolderAction]

limit

The maximum number of results to return per request.

Return type long

class dropbox.sharing.**ListFoldersContinueArg**(*cursor=None*)

Bases: `object`

Variables ***cursor*** – The cursor returned by the previous API call specified in the endpoint description.

__init__(*cursor=None*)

cursor

The cursor returned by the previous API call specified in the endpoint description.

Return type str

class dropbox.sharing.**ListFoldersContinueError**(*tag*, *value=None*)

Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***invalid_cursor*** – `ListFoldersContinueArg.cursor` is invalid.

invalid_cursor = ListFoldersContinueError('invalid_cursor', None)

is_invalid_cursor()

Check if the union tag is `invalid_cursor`.

Return type bool

is_other()

Check if the union tag is `other`.

Return type bool

other = ListFoldersContinueError('other', None)

class dropbox.sharing.**ListFoldersResult**(*entries=None*, *cursor=None*)

Bases: `object`

Result for *dropbox.dropbox.Dropbox.sharing_list_folders()* or *dropbox.dropbox.Dropbox.sharing_list_mountable_folders()*, depending on which endpoint was requested. Unmounted shared folders can be identified by the absence of `SharedFolderMetadata.path_lower`.

Variables

- ***entries*** – List of all shared folders the authenticated user has access to.

- ***cursor*** – Present if there are additional shared folders that have not been returned yet. Pass the cursor into the corresponding continue endpoint (either `dropbox.dropbox.Dropbox.sharing_list_folders_continue()` or `dropbox.dropbox.Dropbox.sharing_list_mountable_folders_continue()`) to list additional folders.

__init__(*entries=None*, *cursor=None*)

cursor

Present if there are additional shared folders that have not been returned yet. Pass the cursor into the corresponding continue endpoint (either `dropbox.dropbox.Dropbox.sharing_list_folders_continue()` or `dropbox.dropbox.Dropbox.sharing_list_mountable_folders_continue()`) to list additional folders.

Return type str

entries

List of all shared folders the authenticated user has access to.

Return type list of [SharedFolderMetadata]

class dropbox.sharing.**ListSharedLinksArg**(*path=None*, *cursor=None*, *direct_only=None*)

Bases: `object`

Variables

- ***path*** – See `dropbox.dropbox.Dropbox.sharing_list_shared_links()` description.
- ***cursor*** – The cursor returned by your last call to `dropbox.dropbox.Dropbox.sharing_list_shared_links()`.
- ***direct_only*** – See `dropbox.dropbox.Dropbox.sharing_list_shared_links()` description.

__init__(*path=None*, *cursor=None*, *direct_only=None*)

cursor

The cursor returned by your last call to `dropbox.dropbox.Dropbox.sharing_list_shared_links()`.

Return type str

direct_only

See `dropbox.dropbox.Dropbox.sharing_list_shared_links()` description.

Return type bool

path

See `dropbox.dropbox.Dropbox.sharing_list_shared_links()` description.

Return type str

class dropbox.sharing.**ListSharedLinksError**(*tag*, *value=None*)

Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***reset*** – Indicates that the cursor has been invalidated. Call `dropbox.dropbox.Dropbox.sharing_list_shared_links()` to obtain a new cursor.

get_path()
Only call this if *is_path()* is true.

Return type files.LookupError_validator

is_other()
Check if the union tag is `other`.

Return type bool

is_path()
Check if the union tag is `path`.

Return type bool

is_reset()
Check if the union tag is `reset`.

Return type bool

other = ListSharedLinksError('other', None)

classmethod path(*val*)
Create an instance of this class set to the `path` tag with value `val`.

Parameters **val** (*files.LookupError_validator*) –

Return type *ListSharedLinksError*

reset = ListSharedLinksError('reset', None)

class dropbox.sharing.**ListSharedLinksResult**(*links=None*, *has_more=None*, *cursor=None*)
Bases: `object`

Variables

- ***links*** – Shared links applicable to the path argument.
- ***has_more*** – Is true if there are additional shared links that have not been returned yet. Pass the cursor into *dropbox.dropbox.Dropbox.sharing_list_shared_links()* to retrieve them.
- ***cursor*** – Pass the cursor into *dropbox.dropbox.Dropbox.sharing_list_shared_links()* to obtain the additional links. Cursor is returned only if no path is given.

__init__(*links=None*, *has_more=None*, *cursor=None*)

cursor
Pass the cursor into *dropbox.dropbox.Dropbox.sharing_list_shared_links()* to obtain the additional links. Cursor is returned only if no path is given.

Return type str

has_more
Is true if there are additional shared links that have not been returned yet. Pass the cursor into *dropbox.dropbox.Dropbox.sharing_list_shared_links()* to retrieve them.

Return type bool

links
Shared links applicable to the path argument.

Return type list of [SharedLinkMetadata]

class dropbox.sharing.**MemberAccessLevelResult**(*access_level=None*, *warning=None*, *access_details=None*)
Bases: `object`

Contains information about a member's access level to content after an operation.

Variables

- ***access_level*** – The member still has this level of access to the content through a parent folder.
- ***warning*** – A localized string with additional information about why the user has this access level to the content.
- ***access_details*** – The parent folders that a member has access to. The field is present if the user has access to the first parent folder where the member gains access.

__init__(*access_level=None*, *warning=None*, *access_details=None*)

access_details

The parent folders that a member has access to. The field is present if the user has access to the first parent folder where the member gains access.

Return type list of [ParentFolderAccessInfo]

access_level

The member still has this level of access to the content through a parent folder.

Return type *AccessLevel*

warning

A localized string with additional information about why the user has this access level to the content.

Return type str

class dropbox.sharing.**MemberAction**(*tag*, *value=None*)

Bases: dropbox.stone_base.Union

Actions that may be taken on members of a shared folder.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***leave_a_copy*** – Allow the member to keep a copy of the folder when removing.
- ***make_editor*** – Make the member an editor of the folder.
- ***make_owner*** – Make the member an owner of the folder.
- ***make_viewer*** – Make the member a viewer of the folder.
- ***make_viewer_no_comment*** – Make the member a viewer of the folder without commenting permissions.
- ***remove*** – Remove the member from the folder.

is_leave_a_copy()

Check if the union tag is `leave_a_copy`.

Return type bool

is_make_editor()

Check if the union tag is `make_editor`.

Return type bool

is_make_owner()

Check if the union tag is `make_owner`.

Return type bool

is_make_viewer()
Check if the union tag is `make_viewer`.

Return type bool

is_make_viewer_no_comment()
Check if the union tag is `make_viewer_no_comment`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_remove()
Check if the union tag is `remove`.

Return type bool

leave_a_copy = MemberAction('leave_a_copy', None)

make_editor = MemberAction('make_editor', None)

make_owner = MemberAction('make_owner', None)

make_viewer = MemberAction('make_viewer', None)

make_viewer_no_comment = MemberAction('make_viewer_no_comment', None)

other = MemberAction('other', None)

remove = MemberAction('remove', None)

class dropbox.sharing.**MemberPermission**(*action=None*, *allow=None*, *reason=None*)
Bases: `object`

Whether the user is allowed to take the action on the associated member.

Variables

- ***action*** – The action that the user may wish to take on the member.
- ***allow*** – True if the user is allowed to take the action.
- ***reason*** – The reason why the user is denied the permission. Not present if the action is allowed

__init__(*action=None*, *allow=None*, *reason=None*)

action
The action that the user may wish to take on the member.

Return type *MemberAction*

allow
True if the user is allowed to take the action.

Return type bool

reason
The reason why the user is denied the permission. Not present if the action is allowed

Return type *PermissionDeniedReason*

class `dropbox.sharing.`**`MemberPolicy`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Policy governing who can be a member of a shared folder. Only applicable to folders owned by a user on a team.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***team*** – Only a teammate can become a member.
- ***anyone*** – Anyone can become a member.

`anyone` = MemberPolicy('anyone', None)

`is_anyone`()
Check if the union tag is `anyone`.

Return type bool

`is_other`()
Check if the union tag is `other`.

Return type bool

`is_team`()
Check if the union tag is `team`.

Return type bool

`other` = MemberPolicy('other', None)

`team` = MemberPolicy('team', None)

class `dropbox.sharing.`**`MemberSelector`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Includes different ways to identify a member of a shared folder.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***dropbox_id*** (*str*) – Dropbox account, team member, or group ID of member.
- ***email*** (*str*) – E-mail address of member.

classmethod `dropbox_id`(*val*)
Create an instance of this class set to the `dropbox_id` tag with value `val`.

Parameters **`val`** (*str*) –

Return type *MemberSelector*

classmethod `email`(*val*)
Create an instance of this class set to the `email` tag with value `val`.

Parameters **`val`** (*str*) –

Return type *MemberSelector*

`get_dropbox_id`()
Dropbox account, team member, or group ID of member.

Only call this if *`is_dropbox_id()`* is true.

Return type str

get_email()
E-mail address of member.

Only call this if *is_email()* is true.

Return type str

is_dropbox_id()
Check if the union tag is `dropbox_id`.

Return type bool

is_email()
Check if the union tag is `email`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

other = MemberSelector('other', None)

class dropbox.sharing.**MembershipInfo**(*access_type=None*, *permissions=None*, *initials=None*, *is_inherited=None*)

Bases: `object`

The information about a member of the shared content.

Variables

- ***access_type*** – The access type for this member.
- ***permissions*** – The permissions that requesting user has on this member. The set of permissions corresponds to the MemberActions in the request.
- ***initials*** – Suggested name initials for a member.
- ***is_inherited*** – True if the member has access from a parent folder.

__init__(*access_type=None*, *permissions=None*, *initials=None*, *is_inherited=None*)

access_type
The access type for this member.

Return type *AccessLevel*

initials
Suggested name initials for a member.

Return type str

is_inherited
True if the member has access from a parent folder.

Return type bool

permissions
The permissions that requesting user has on this member. The set of permissions corresponds to the MemberActions in the request.

Return type list of [MemberPermission]

class dropbox.sharing.**ModifySharedLinkSettingsArgs**(*url=None*, *settings=None*, *remove_expiration=None*)

Bases: object

Variables

- ***url*** – URL of the shared link to change its settings
- ***settings*** – Set of settings for the shared link.
- ***remove_expiration*** – If set to true, removes the expiration of the shared link.

__init__(*url=None*, *settings=None*, *remove_expiration=None*)

remove_expiration

If set to true, removes the expiration of the shared link.

Return type bool

settings

Set of settings for the shared link.

Return type *SharedLinkSettings*

url

URL of the shared link to change its settings

Return type str

class dropbox.sharing.**ModifySharedLinkSettingsError**(*tag*, *value=None*)

Bases: *dropbox.sharing.SharedLinkError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***settings_error*** (SharedLinkSettingsError) – There is an error with the given settings
- ***email_not_verified*** – The caller's email should be verified

email_not_verified = ModifySharedLinkSettingsError('email_not_verified', None)

get_settings_error()

There is an error with the given settings

Only call this if *is_settings_error()* is true.

Return type *SharedLinkSettingsError*

is_email_not_verified()

Check if the union tag is `email_not_verified`.

Return type bool

is_settings_error()

Check if the union tag is `settings_error`.

Return type bool

classmethod settings_error(*val*)

Create an instance of this class set to the `settings_error` tag with value `val`.

Parameters **val** (SharedLinkSettingsError) –

Return type *ModifySharedLinkSettingsError*

class `dropbox.sharing.`**`MountFolderArg`**(*shared_folder_id=None*)
Bases: `object`

Variables ***shared_folder_id*** – The ID of the shared folder to mount.

`__init__`(*shared_folder_id=None*)

`shared_folder_id`
The ID of the shared folder to mount.

Return type str

class `dropbox.sharing.`**`MountFolderError`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***inside_shared_folder*** – Mounting would cause a shared folder to be inside another, which is disallowed.
- ***insufficient_quota*** (`InsufficientQuotaAmounts`) – The current user does not have enough space to mount the shared folder.
- ***already_mounted*** – The shared folder is already mounted.
- ***no_permission*** – The current user does not have permission to perform this action.
- ***not_mountable*** – The shared folder is not mountable. One example where this can occur is when the shared folder belongs within a team folder in the user's Dropbox.

classmethod `access_error`(*val*)
Create an instance of this class set to the `access_error` tag with value `val`.

Parameters **`val`** (`SharedFolderAccessError`) –

Return type *MountFolderError*

`already_mounted` = MountFolderError('already_mounted', None)

`get_access_error`()
Only call this if *`is_access_error()`* is true.

Return type *SharedFolderAccessError*

`get_insufficient_quota`()
The current user does not have enough space to mount the shared folder.

Only call this if *`is_insufficient_quota()`* is true.

Return type *InsufficientQuotaAmounts*

`inside_shared_folder` = MountFolderError('inside_shared_folder', None)

classmethod `insufficient_quota`(*val*)
Create an instance of this class set to the `insufficient_quota` tag with value `val`.

Parameters **`val`** (`InsufficientQuotaAmounts`) –

Return type *MountFolderError*

`is_access_error`()
Check if the union tag is `access_error`.

Return type bool

is_already_mounted()
Check if the union tag is `already_mounted`.

Return type bool

is_inside_shared_folder()
Check if the union tag is `inside_shared_folder`.

Return type bool

is_insufficient_quota()
Check if the union tag is `insufficient_quota`.

Return type bool

is_no_permission()
Check if the union tag is `no_permission`.

Return type bool

is_not_mountable()
Check if the union tag is `not_mountable`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

no_permission = MountFolderError('no_permission', None)

not_mountable = MountFolderError('not_mountable', None)

other = MountFolderError('other', None)

class dropbox.sharing.**ParentFolderAccessInfo**(*folder_name=None*, *shared_folder_id=None*, *permissions=None*)

Bases: `object`

Contains information about a parent folder that a member has access to.

Variables

- ***folder_name*** – Display name for the folder.
- ***shared_folder_id*** – The identifier of the parent shared folder.
- ***permissions*** – The user's permissions for the parent shared folder.

__init__(*folder_name=None*, *shared_folder_id=None*, *permissions=None*)

folder_name
Display name for the folder.

Return type str

permissions
The user's permissions for the parent shared folder.

Return type list of [MemberPermission]

shared_folder_id
The identifier of the parent shared folder.

Return type str

class `dropbox.sharing.`**`PathLinkMetadata`**(*url=None*, *visibility=None*, *path=None*, *expires=None*)
Bases: `dropbox.sharing.LinkMetadata`

Metadata for a path-based shared link.

Variables ***path*** – Path in user's Dropbox.

`__init__`(*url=None*, *visibility=None*, *path=None*, *expires=None*)

`path`
Path in user's Dropbox.

Return type str

class `dropbox.sharing.`**`PendingUploadMode`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Flag to indicate pending upload default (for linking to not-yet-existing paths).

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***file*** – Assume pending uploads are files.
- ***folder*** – Assume pending uploads are folders.

`file` = PendingUploadMode('file', None)

`folder` = PendingUploadMode('folder', None)

`is_file`()
Check if the union tag is `file`.

Return type bool

`is_folder`()
Check if the union tag is `folder`.

Return type bool

class `dropbox.sharing.`**`PermissionDeniedReason`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Possible reasons the user is denied a permission.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***user_not_same_team_as_owner*** – User is not on the same team as the folder owner.
- ***user_not_allowed_by_owner*** – User is prohibited by the owner from taking the action.
- ***target_is_indirect_member*** – Target is indirectly a member of the folder, for example by being part of a group.
- ***target_is_owner*** – Target is the owner of the folder.
- ***target_is_self*** – Target is the user itself.
- ***target_not_active*** – Target is not an active member of the team.
- ***folder_is_limited_team_folder*** – Folder is team folder for a limited team.

folder_is_limited_team_folder = PermissionDeniedReason('folder_is_limited_team_folder', None)

is_folder_is_limited_team_folder()
Check if the union tag is `folder_is_limited_team_folder`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_target_is_indirect_member()
Check if the union tag is `target_is_indirect_member`.

Return type bool

is_target_is_owner()
Check if the union tag is `target_is_owner`.

Return type bool

is_target_is_self()
Check if the union tag is `target_is_self`.

Return type bool

is_target_not_active()
Check if the union tag is `target_not_active`.

Return type bool

is_user_not_allowed_by_owner()
Check if the union tag is `user_not_allowed_by_owner`.

Return type bool

is_user_not_same_team_as_owner()
Check if the union tag is `user_not_same_team_as_owner`.

Return type bool

other = PermissionDeniedReason('other', None)

target_is_indirect_member = PermissionDeniedReason('target_is_indirect_member', None)

target_is_owner = PermissionDeniedReason('target_is_owner', None)

target_is_self = PermissionDeniedReason('target_is_self', None)

target_not_active = PermissionDeniedReason('target_not_active', None)

user_not_allowed_by_owner = PermissionDeniedReason('user_not_allowed_by_owner', None)

user_not_same_team_as_owner = PermissionDeniedReason('user_not_same_team_as_owner', None)

class dropbox.sharing.**RelinquishFileMembershipArg**(*file=None*)
Bases: `object`

Variables ***file*** – The path or id for the file.

__init__(*file=None*)

file
The path or id for the file.

Return type str

class dropbox.sharing.**RelinquishFileMembershipError**(*tag*, *value=None*)

Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables

- ***group_access*** – The current user has access to the shared file via a group. You can't relinquish membership to a file shared via groups.
- ***no_permission*** – The current user does not have permission to perform this action.

classmethod access_error(*val*)

Create an instance of this class set to the access_error tag with value val.

Parameters **val** (SharingFileAccessError) –

Return type *RelinquishFileMembershipError*

get_access_error()

Only call this if *is_access_error()* is true.

Return type *SharingFileAccessError*

group_access = RelinquishFileMembershipError('group_access', None)

is_access_error()

Check if the union tag is access_error.

Return type bool

is_group_access()

Check if the union tag is group_access.

Return type bool

is_no_permission()

Check if the union tag is no_permission.

Return type bool

is_other()

Check if the union tag is other.

Return type bool

no_permission = RelinquishFileMembershipError('no_permission', None)

other = RelinquishFileMembershipError('other', None)

class dropbox.sharing.**RelinquishFolderMembershipArg**(*shared_folder_id=None*, *leave_a_copy=None*)

Bases: object

Variables

- ***shared_folder_id*** – The ID for the shared folder.
- ***leave_a_copy*** – Keep a copy of the folder's contents upon relinquishing membership.

__init__(*shared_folder_id=None*, *leave_a_copy=None*)

leave_a_copy

Keep a copy of the folder's contents upon relinquishing membership.

Return type bool

shared_folder_id
The ID for the shared folder.

Return type str

class dropbox.sharing.**RelinquishFolderMembershipError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables

- ***folder_owner*** – The current user is the owner of the shared folder. Owners cannot relinquish membership to their own folders. Try unsharing or transferring ownership first.
- ***mounted*** – The shared folder is currently mounted. Unmount the shared folder before relinquishing membership.
- ***group_access*** – The current user has access to the shared folder via a group. You can't relinquish membership to folders shared via groups.
- ***team_folder*** – This action cannot be performed on a team shared folder.
- ***no_permission*** – The current user does not have permission to perform this action.

classmethod access_error(*val*)
Create an instance of this class set to the access_error tag with value val.

Parameters **val** (SharedFolderAccessError) –

Return type *RelinquishFolderMembershipError*

folder_owner = RelinquishFolderMembershipError('folder_owner', None)

get_access_error()
Only call this if *is_access_error()* is true.

Return type *SharedFolderAccessError*

group_access = RelinquishFolderMembershipError('group_access', None)

is_access_error()
Check if the union tag is access_error.

Return type bool

is_folder_owner()
Check if the union tag is folder_owner.

Return type bool

is_group_access()
Check if the union tag is group_access.

Return type bool

is_mounted()
Check if the union tag is mounted.

Return type bool

is_no_permission()
Check if the union tag is no_permission.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_team_folder()
Check if the union tag is `team_folder`.

Return type bool

mounted = RelinquishFolderMembershipError('mounted', None)

no_permission = RelinquishFolderMembershipError('no_permission', None)

other = RelinquishFolderMembershipError('other', None)

team_folder = RelinquishFolderMembershipError('team_folder', None)

class dropbox.sharing.**RemoveFileMemberArg**(*file=None*, *member=None*)
Bases: `object`

Arguments for *dropbox.dropbox.Dropbox.sharing_remove_file_member_2()*.

Variables

- ***file*** – File from which to remove members.
- ***member*** – Member to remove from this file. Note that even if an email is specified, it may result in the removal of a user (not an invitee) if the user's main account corresponds to that email address.

__init__(*file=None*, *member=None*)

file
File from which to remove members.

Return type str

member
Member to remove from this file. Note that even if an email is specified, it may result in the removal of a user (not an invitee) if the user's main account corresponds to that email address.

Return type *MemberSelector*

class dropbox.sharing.**RemoveFileMemberError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Errors for *dropbox.dropbox.Dropbox.sharing_remove_file_member_2()*.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***no_explicit_access*** (`MemberAccessLevelResult`) – This member does not have explicit access to the file and therefore cannot be removed. The return value is the access that a user might have to the file from a parent folder.

classmethod access_error(*val*)
Create an instance of this class set to the `access_error` tag with value `val`.

Parameters **val** (`SharingFileAccessError`) –

Return type *RemoveFileMemberError*

get_access_error()
Only call this if *is_access_error()* is true.

Return type *SharingFileAccessError*

get_no_explicit_access()
This member does not have explicit access to the file and therefore cannot be removed. The return value is the access that a user might have to the file from a parent folder.

Only call this if `is_no_explicit_access()` is true.

Return type *MemberAccessLevelResult*

get_user_error()
Only call this if `is_user_error()` is true.

Return type *SharingUserError*

is_access_error()
Check if the union tag is `access_error`.

Return type bool

is_no_explicit_access()
Check if the union tag is `no_explicit_access`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_user_error()
Check if the union tag is `user_error`.

Return type bool

classmethod no_explicit_access(*val*)
Create an instance of this class set to the `no_explicit_access` tag with value `val`.

Parameters **val** (`MemberAccessLevelResult`) –

Return type *RemoveFileMemberError*

other = RemoveFileMemberError('other', None)

classmethod user_error(*val*)
Create an instance of this class set to the `user_error` tag with value `val`.

Parameters **val** (`SharingUserError`) –

Return type *RemoveFileMemberError*

class dropbox.sharing.**RemoveFolderMemberArg**(*shared_folder_id=None*, *member=None*, *leave_a_copy=None*)

Bases: `object`

Variables

- ***shared_folder_id*** – The ID for the shared folder.
- ***member*** – The member to remove from the folder.
- ***leave_a_copy*** – If true, the removed user will keep their copy of the folder after it's unshared, assuming it was mounted. Otherwise, it will be removed from their Dropbox. Also, this must be set to false when kicking a group.

__init__(*shared_folder_id=None*, *member=None*, *leave_a_copy=None*)

leave_a_copy
If true, the removed user will keep their copy of the folder after it's unshared, assuming it was mounted. Otherwise, it will be removed from their Dropbox. Also, this must be set to false when kicking a group.

Return type bool

member
The member to remove from the folder.

Return type *MemberSelector*

shared_folder_id
The ID for the shared folder.

Return type str

class `dropbox.sharing.`**RemoveFolderMemberError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***folder_owner*** – The target user is the owner of the shared folder. You can't remove this user until ownership has been transferred to another member.
- ***group_access*** – The target user has access to the shared folder via a group.
- ***team_folder*** – This action cannot be performed on a team shared folder.
- ***no_permission*** – The current user does not have permission to perform this action.

classmethod access_error(*val*)
Create an instance of this class set to the `access_error` tag with value `val`.

Parameters **val** (`SharedFolderAccessError`) –

Return type *RemoveFolderMemberError*

folder_owner = RemoveFolderMemberError('folder_owner', None)

get_access_error()
Only call this if *`is_access_error()`* is true.

Return type *SharedFolderAccessError*

get_member_error()
Only call this if *`is_member_error()`* is true.

Return type *SharedFolderMemberError*

group_access = RemoveFolderMemberError('group_access', None)

is_access_error()
Check if the union tag is `access_error`.

Return type bool

is_folder_owner()
Check if the union tag is `folder_owner`.

Return type bool

is_group_access()
Check if the union tag is `group_access`.

Return type bool

is_member_error()
Check if the union tag is `member_error`.

Return type bool

is_no_permission()
Check if the union tag is `no_permission`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_team_folder()
Check if the union tag is `team_folder`.

Return type bool

classmethod member_error(*val*)
Create an instance of this class set to the `member_error` tag with value `val`.

Parameters **val** (`SharedFolderMemberError`) –

Return type *RemoveFolderMemberError*

no_permission = RemoveFolderMemberError('no_permission', None)

other = RemoveFolderMemberError('other', None)

team_folder = RemoveFolderMemberError('team_folder', None)

class dropbox.sharing.**RemoveMemberJobStatus**(*tag*, *value=None*)
Bases: *dropbox.async.PollResultBase*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***complete*** (`MemberAccessLevelResult`) – Removing the folder member has finished. The value is information about whether the member has another form of access.

classmethod complete(*val*)
Create an instance of this class set to the `complete` tag with value `val`.

Parameters **val** (`MemberAccessLevelResult`) –

Return type *RemoveMemberJobStatus*

classmethod failed(*val*)
Create an instance of this class set to the `failed` tag with value `val`.

Parameters **val** (`RemoveFolderMemberError`) –

Return type *RemoveMemberJobStatus*

get_complete()
Removing the folder member has finished. The value is information about whether the member has another form of access.

Only call this if *is_complete()* is true.

Return type *MemberAccessLevelResult*

get_failed()
Only call this if *is_failed()* is true.

Return type *RemoveFolderMemberError*

is_complete()
Check if the union tag is `complete`.

Return type bool

is_failed()
Check if the union tag is `failed`.

Return type bool

class dropbox.sharing.**RequestedVisibility**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

The access permission that can be requested by the caller for the shared link. Note that the final resolved visibility of the shared link takes into account other aspects, such as team and shared folder settings. Check the *ResolvedVisibility* for more info on the possible resolved visibility values of shared links.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***public*** – Anyone who has received the link can access it. No login required.
- ***team_only*** – Only members of the same team can access the link. Login is required.
- ***password*** – A link-specific password is required to access the link. Login is not required.

is_password()
Check if the union tag is `password`.

Return type bool

is_public()
Check if the union tag is `public`.

Return type bool

is_team_only()
Check if the union tag is `team_only`.

Return type bool

password = RequestedVisibility('password', None)

public = RequestedVisibility('public', None)

team_only = RequestedVisibility('team_only', None)

class dropbox.sharing.**ResolvedVisibility**(*tag*, *value=None*)
Bases: *dropbox.sharing.RequestedVisibility*

The actual access permissions values of shared links after taking into account user preferences and the team and shared folder settings. Check the *RequestedVisibility* for more info on the possible visibility values that can be set by the shared link's owner.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***team_and_password*** – Only members of the same team who have the link-specific password can access the link. Login is required.
- ***shared_folder_only*** – Only members of the shared folder containing the linked file can access the link. Login is required.

is_other()
Check if the union tag is `other`.

Return type bool

is_shared_folder_only()
Check if the union tag is `shared_folder_only`.

Return type bool

is_team_and_password()
Check if the union tag is `team_and_password`.

Return type bool

other = ResolvedVisibility('other', None)

shared_folder_only = ResolvedVisibility('shared_folder_only', None)

team_and_password = ResolvedVisibility('team_and_password', None)

class dropbox.sharing.**RevokeSharedLinkArg**(*url=None*)
Bases: `object`

Variables ***url*** – URL of the shared link.

__init__(*url=None*)

url
URL of the shared link.

Return type str

class dropbox.sharing.**RevokeSharedLinkError**(*tag*, *value=None*)
Bases: *dropbox.sharing.SharedLinkError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***shared_link_malformed*** – Shared link is malformed.

is_shared_link_malformed()
Check if the union tag is `shared_link_malformed`.

Return type bool

shared_link_malformed = RevokeSharedLinkError('shared_link_malformed', None)

class dropbox.sharing.**ShareFolderArg**(*path=None*, *member_policy=None*, *acl_update_policy=None*, *shared_link_policy=None*, *force_async=None*)
Bases: `object`

Variables

- ***path*** – The path to the folder to share. If it does not exist, then a new one is created.
- ***member_policy*** – Who can be a member of this shared folder. Only applicable if the current user is on a team.
- ***acl_update_policy*** – Who can add and remove members of this shared folder.

- ***shared_link_policy*** – The policy to apply to shared links created for content inside this shared folder. The current user must be on a team to set this policy to `SharedLinkPolicy.members`.
- ***force_async*** – Whether to force the share to happen asynchronously.

__init__(*path=None*, *member_policy=None*, *acl_update_policy=None*, *shared_link_policy=None*, *force_async=None*)

acl_update_policy
Who can add and remove members of this shared folder.

Return type *AclUpdatePolicy*

force_async
Whether to force the share to happen asynchronously.

Return type bool

member_policy
Who can be a member of this shared folder. Only applicable if the current user is on a team.

Return type *MemberPolicy*

path
The path to the folder to share. If it does not exist, then a new one is created.

Return type str

shared_link_policy
The policy to apply to shared links created for content inside this shared folder. The current user must be on a team to set this policy to `SharedLinkPolicy.members`.

Return type *SharedLinkPolicy*

class dropbox.sharing.**ShareFolderError**(*tag*, *value=None*)
Bases: *dropbox.sharing.ShareFolderErrorBase*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***no_permission*** – The current user does not have permission to perform this action.

is_no_permission()
Check if the union tag is `no_permission`.

Return type bool

no_permission = ShareFolderError('no_permission', None)

class dropbox.sharing.**ShareFolderErrorBase**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***email_unverified*** – The current user's e-mail address is unverified.
- ***bad_path*** (`SharePathError`) – `ShareFolderArg.path` is invalid.
- ***team_policy_disallows_member_policy*** – Team policy is more restrictive than `ShareFolderArg.member_policy`.
- ***disallowed_shared_link_policy*** – The current user's account is not allowed to select the specified `ShareFolderArg.shared_link_policy`.

classmethod bad_path(*val*)
Create an instance of this class set to the `bad_path` tag with value `val`.

Parameters **val** (`SharePathError`) –

Return type *ShareFolderErrorBase*

disallowed_shared_link_policy = ShareFolderErrorBase('disallowed_shared_link_policy', None)

email_unverified = ShareFolderErrorBase('email_unverified', None)

get_bad_path()
`ShareFolderArg.path` is invalid.

Only call this if *`is_bad_path()`* is true.

Return type *SharePathError*

is_bad_path()
Check if the union tag is `bad_path`.

Return type bool

is_disallowed_shared_link_policy()
Check if the union tag is `disallowed_shared_link_policy`.

Return type bool

is_email_unverified()
Check if the union tag is `email_unverified`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_team_policy_disallows_member_policy()
Check if the union tag is `team_policy_disallows_member_policy`.

Return type bool

other = ShareFolderErrorBase('other', None)

team_policy_disallows_member_policy = ShareFolderErrorBase('team_policy_disallows_member_policy', Non

class `dropbox.sharing.`**ShareFolderJobStatus**(*tag*, *value=None*)
Bases: *`dropbox.async.PollResultBase`*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***complete*** (`SharedFolderMetadata`) – The share job has finished. The value is the metadata for the folder.

classmethod complete(*val*)
Create an instance of this class set to the `complete` tag with value `val`.

Parameters **val** (`SharedFolderMetadata`) –

Return type *ShareFolderJobStatus*

classmethod failed(*val*)
Create an instance of this class set to the `failed` tag with value `val`.

Parameters **val** (`ShareFolderError`) –

Return type *ShareFolderJobStatus*

get_complete()
The share job has finished. The value is the metadata for the folder.

Only call this if `is_complete()` is true.

Return type *SharedFolderMetadata*

get_failed()
Only call this if `is_failed()` is true.

Return type *ShareFolderError*

is_complete()
Check if the union tag is `complete`.

Return type bool

is_failed()
Check if the union tag is `failed`.

Return type bool

class `dropbox.sharing.`**ShareFolderLaunch**(*tag*, *value=None*)
Bases: `dropbox.async.LaunchResultBase`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

classmethod complete(*val*)
Create an instance of this class set to the `complete` tag with value `val`.

Parameters **val** (`SharedFolderMetadata`) –

Return type *ShareFolderLaunch*

get_complete()
Only call this if `is_complete()` is true.

Return type *SharedFolderMetadata*

is_complete()
Check if the union tag is `complete`.

Return type bool

class `dropbox.sharing.`**SharePathError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***is_file*** – A file is at the specified path.
- ***inside_shared_folder*** – We do not support sharing a folder inside a shared folder.
- ***contains_shared_folder*** – We do not support shared folders that contain shared folders.
- ***contains_app_folder*** – We do not support shared folders that contain app folders.
- ***contains_team_folder*** – We do not support shared folders that contain team folders.
- ***is_app_folder*** – We do not support sharing an app folder.

- ***inside_app_folder*** – We do not support sharing a folder inside an app folder.
- ***is_public_folder*** – A public folder can't be shared this way. Use a public link instead.
- ***inside_public_folder*** – A folder inside a public folder can't be shared this way. Use a public link instead.
- ***already_shared*** (`SharedFolderMetadata`) – Folder is already shared. Contains metadata about the existing shared folder.
- ***invalid_path*** – Path is not valid.
- ***is_osx_package*** – We do not support sharing a Mac OS X package.
- ***inside_osx_package*** – We do not support sharing a folder inside a Mac OS X package.
- ***invalid_path_root*** (`PathRootError`) – The path root parameter provided is invalid.

classmethod `already_shared`(*val*)

Create an instance of this class set to the `already_shared` tag with value `val`.

Parameters **`val`** (`SharedFolderMetadata`) –

Return type *SharePathError*

`contains_app_folder` = SharePathError('contains_app_folder', None)

`contains_shared_folder` = SharePathError('contains_shared_folder', None)

`contains_team_folder` = SharePathError('contains_team_folder', None)

`get_already_shared`()

Folder is already shared. Contains metadata about the existing shared folder.

Only call this if `is_already_shared()` is true.

Return type *SharedFolderMetadata*

`get_invalid_path_root`()

The path root parameter provided is invalid.

Only call this if `is_invalid_path_root()` is true.

Return type files.PathRootError_validator

`inside_app_folder` = SharePathError('inside_app_folder', None)

`inside_osx_package` = SharePathError('inside_osx_package', None)

`inside_public_folder` = SharePathError('inside_public_folder', None)

`inside_shared_folder` = SharePathError('inside_shared_folder', None)

`invalid_path` = SharePathError('invalid_path', None)

classmethod `invalid_path_root`(*val*)

Create an instance of this class set to the `invalid_path_root` tag with value `val`.

Parameters **`val`** (`files.PathRootError_validator`) –

Return type *SharePathError*

`is_already_shared`()

Check if the union tag is `already_shared`.

Return type bool

`is_app_folder` = SharePathError('is_app_folder', None)

is_contains_app_folder()
Check if the union tag is `contains_app_folder`.

Return type bool

is_contains_shared_folder()
Check if the union tag is `contains_shared_folder`.

Return type bool

is_contains_team_folder()
Check if the union tag is `contains_team_folder`.

Return type bool

is_file = SharePathError('is_file', None)

is_inside_app_folder()
Check if the union tag is `inside_app_folder`.

Return type bool

is_inside_osx_package()
Check if the union tag is `inside_osx_package`.

Return type bool

is_inside_public_folder()
Check if the union tag is `inside_public_folder`.

Return type bool

is_inside_shared_folder()
Check if the union tag is `inside_shared_folder`.

Return type bool

is_invalid_path()
Check if the union tag is `invalid_path`.

Return type bool

is_invalid_path_root()
Check if the union tag is `invalid_path_root`.

Return type bool

is_is_app_folder()
Check if the union tag is `is_app_folder`.

Return type bool

is_is_file()
Check if the union tag is `is_file`.

Return type bool

is_is_osx_package()
Check if the union tag is `is_osx_package`.

Return type bool

is_is_public_folder()
Check if the union tag is `is_public_folder`.

Return type bool

is_osx_package = SharePathError('is_osx_package', None)

is_other()

Check if the union tag is `other`.

Return type bool

is_public_folder = SharePathError('is_public_folder', None)

other = SharePathError('other', None)

class dropbox.sharing.**SharedFileMembers**(*users=None*, *groups=None*, *invitees=None*, *cursor=None*)

Bases: `object`

Shared file user, group, and invitee membership. Used for the results of `dropbox.dropbox.Dropbox.sharing_list_file_members()` and `dropbox.dropbox.Dropbox.sharing_list_file_members_continue()`, and used as part of the results for `dropbox.dropbox.Dropbox.sharing_list_file_members_batch()`.

Variables

- ***users*** – The list of user members of the shared file.
- ***groups*** – The list of group members of the shared file.
- ***invitees*** – The list of invited members of a file, but have not logged in and claimed this.
- ***cursor*** – Present if there are additional shared file members that have not been returned yet. Pass the cursor into `dropbox.dropbox.Dropbox.sharing_list_file_members_continue()` to list additional members.

__init__(*users=None*, *groups=None*, *invitees=None*, *cursor=None*)

cursor

Present if there are additional shared file members that have not been returned yet. Pass the cursor into `dropbox.dropbox.Dropbox.sharing_list_file_members_continue()` to list additional members.

Return type str

groups

The list of group members of the shared file.

Return type list of [GroupMembershipInfo]

invitees

The list of invited members of a file, but have not logged in and claimed this.

Return type list of [InviteeMembershipInfo]

users

The list of user members of the shared file.

Return type list of [UserMembershipInfo]

class dropbox.sharing.**SharedFileMetadata**(*policy=None*, *preview_url=None*, *name=None*, *id=None*, *permissions=None*, *owner_team=None*, *parent_shared_folder_id=None*, *path_lower=None*, *path_display=None*, *time_invited=None*)

Bases: `object`

Properties of the shared file.

Variables

- ***policy*** – Policies governing this shared file.
- ***permissions*** – The sharing permissions that requesting user has on this file. This corresponds to the entries given in `GetFileMetadataBatchArg.actions` or `GetFileMetadataArg.actions`.
- ***owner_team*** – The team that owns the file. This field is not present if the file is not owned by a team.
- ***parent_shared_folder_id*** – The ID of the parent shared folder. This field is present only if the file is contained within a shared folder.
- ***preview_url*** – URL for displaying a web preview of the shared file.
- ***path_lower*** – The lower-case full path of this file. Absent for unmounted files.
- ***path_display*** – The cased path to be used for display purposes only. In rare instances the casing will not correctly match the user's filesystem, but this behavior will match the path provided in the Core API v1. Absent for unmounted files.
- ***name*** – The name of this file.
- ***id*** – The ID of the file.
- ***time_invited*** – Timestamp indicating when the current user was invited to this shared file. If the user was not invited to the shared file, the timestamp will indicate when the user was invited to the parent shared folder. This value may be absent.

__init__(*policy=None*, *preview_url=None*, *name=None*, *id=None*, *permissions=None*, *owner_team=None*, *parent_shared_folder_id=None*, *path_lower=None*, *path_display=None*, *time_invited=None*)

id

The ID of the file.

Return type str

name

The name of this file.

Return type str

owner_team

The team that owns the file. This field is not present if the file is not owned by a team.

Return type users.Team_validator

parent_shared_folder_id

The ID of the parent shared folder. This field is present only if the file is contained within a shared folder.

Return type str

path_display

The cased path to be used for display purposes only. In rare instances the casing will not correctly match the user's filesystem, but this behavior will match the path provided in the Core API v1. Absent for unmounted files.

Return type str

path_lower

The lower-case full path of this file. Absent for unmounted files.

Return type str

permissions
The sharing permissions that requesting user has on this file. This corresponds to the entries given in `GetFileMetadataBatchArg.actions` or `GetFileMetadataArg.actions`.

Return type list of [FilePermission]

policy
Policies governing this shared file.

Return type *FolderPolicy*

preview_url
URL for displaying a web preview of the shared file.

Return type str

time_invited
Timestamp indicating when the current user was invited to this shared file. If the user was not invited to the shared file, the timestamp will indicate when the user was invited to the parent shared folder. This value may be absent.

Return type datetime.datetime

class `dropbox.sharing.`**`SharedFolderAccessError`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

There is an error accessing the shared folder.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***invalid_id*** – This shared folder ID is invalid.
- ***not_a_member*** – The user is not a member of the shared folder thus cannot access it.
- ***email_unverified*** – The current user's e-mail address is unverified.
- ***unmounted*** – The shared folder is unmounted.

email_unverified = SharedFolderAccessError('email_unverified', None)

invalid_id = SharedFolderAccessError('invalid_id', None)

is_email_unverified()
Check if the union tag is `email_unverified`.

Return type bool

is_invalid_id()
Check if the union tag is `invalid_id`.

Return type bool

is_not_a_member()
Check if the union tag is `not_a_member`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_unmounted()
Check if the union tag is `unmounted`.

Return type bool

not_a_member = SharedFolderAccessError('not_a_member', None)

other = SharedFolderAccessError('other', None)

unmounted = SharedFolderAccessError('unmounted', None)

class dropbox.sharing.**SharedFolderMemberError**(*tag*, *value=None*)

Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables

- ***invalid_dropbox_id*** – The target dropbox_id is invalid.
- ***not_a_member*** – The target dropbox_id is not a member of the shared folder.
- ***no_explicit_access*** (MemberAccessLevelResult) – The target member only has inherited access to the shared folder.

get_no_explicit_access()

The target member only has inherited access to the shared folder.

Only call this if *is_no_explicit_access()* is true.

Return type *MemberAccessLevelResult*

invalid_dropbox_id = SharedFolderMemberError('invalid_dropbox_id', None)

is_invalid_dropbox_id()

Check if the union tag is invalid_dropbox_id.

Return type bool

is_no_explicit_access()

Check if the union tag is no_explicit_access.

Return type bool

is_not_a_member()

Check if the union tag is not_a_member.

Return type bool

is_other()

Check if the union tag is other.

Return type bool

classmethod no_explicit_access(*val*)

Create an instance of this class set to the no_explicit_access tag with value val.

Parameters **val** (MemberAccessLevelResult) –

Return type *SharedFolderMemberError*

not_a_member = SharedFolderMemberError('not_a_member', None)

other = SharedFolderMemberError('other', None)

class dropbox.sharing.**SharedFolderMembers**(*users=None*, *groups=None*, *invitees=None*, *cursor=None*)

Bases: object

Shared folder user and group membership.

Variables

- ***users*** – The list of user members of the shared folder.
- ***groups*** – The list of group members of the shared folder.
- ***invitees*** – The list of invitees to the shared folder.
- ***cursor*** – Present if there are additional shared folder members that have not been returned yet. Pass the cursor into `dropbox.dropbox.Dropbox.sharing_list_folder_members_continue()` to list additional members.

__init__(*users=None*, *groups=None*, *invitees=None*, *cursor=None*)

cursor

Present if there are additional shared folder members that have not been returned yet. Pass the cursor into `dropbox.dropbox.Dropbox.sharing_list_folder_members_continue()` to list additional members.

Return type str

groups

The list of group members of the shared folder.

Return type list of [GroupMembershipInfo]

invitees

The list of invitees to the shared folder.

Return type list of [InviteeMembershipInfo]

users

The list of user members of the shared folder.

Return type list of [UserMembershipInfo]

class dropbox.sharing.**SharedFolderMetadata**(*access_type=None*, *is_team_folder=None*, *policy=None*, *name=None*, *shared_folder_id=None*, *time_invited=None*, *preview_url=None*, *owner_team=None*, *parent_shared_folder_id=None*, *path_lower=None*, *permissions=None*)

Bases: `dropbox.sharing.SharedFolderMetadataBase`

The metadata which includes basic information about the shared folder.

Variables

- ***path_lower*** – The lower-cased full path of this shared folder. Absent for unmounted folders.
- ***name*** – The name of the this shared folder.
- ***shared_folder_id*** – The ID of the shared folder.
- ***permissions*** – Actions the current user may perform on the folder and its contents. The set of permissions corresponds to the FolderActions in the request.
- ***time_invited*** – Timestamp indicating when the current user was invited to this shared folder.
- ***preview_url*** – URL for displaying a web preview of the shared folder.

__init__(*access_type=None*, *is_team_folder=None*, *policy=None*, *name=None*, *shared_folder_id=None*, *time_invited=None*, *preview_url=None*, *owner_team=None*, *parent_shared_folder_id=None*, *path_lower=None*, *permissions=None*)

name

The name of the this shared folder.

Return type str

path_lower

The lower-cased full path of this shared folder. Absent for unmounted folders.

Return type str

permissions

Actions the current user may perform on the folder and its contents. The set of permissions corresponds to the FolderActions in the request.

Return type list of [FolderPermission]

preview_url

URL for displaying a web preview of the shared folder.

Return type str

shared_folder_id

The ID of the shared folder.

Return type str

time_invited

Timestamp indicating when the current user was invited to this shared folder.

Return type datetime.datetime

class dropbox.sharing.**SharedFolderMetadataBase**(*access_type=None*, *is_team_folder=None*, *policy=None*, *owner_team=None*, *parent_shared_folder_id=None*)

Bases: object

Properties of the shared folder.

Variables

- ***access_type*** – The current user's access level for this shared folder.
- ***is_team_folder*** – Whether this folder is a team folder.
- ***policy*** – Policies governing this shared folder.
- ***owner_team*** – The team that owns the folder. This field is not present if the folder is not owned by a team.
- ***parent_shared_folder_id*** – The ID of the parent shared folder. This field is present only if the folder is contained within another shared folder.

__init__(*access_type=None*, *is_team_folder=None*, *policy=None*, *owner_team=None*, *parent_shared_folder_id=None*)

access_type

The current user's access level for this shared folder.

Return type *AccessLevel*

is_team_folder

Whether this folder is a team folder.

Return type bool

owner_team
The team that owns the folder. This field is not present if the folder is not owned by a team.

Return type users.Team_validator

parent_shared_folder_id
The ID of the parent shared folder. This field is present only if the folder is contained within another shared folder.

Return type str

policy
Policies governing this shared folder.

Return type *FolderPolicy*

class dropbox.sharing.**SharedLinkAccessFailureReason**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables

- ***login_required*** – User is not logged in.
- ***email_verify_required*** – User's email is not verified.
- ***password_required*** – The link is password protected.
- ***team_only*** – Access is allowed for team members only.
- ***owner_only*** – Access is allowed for the shared link's owner only.

email_verify_required = SharedLinkAccessFailureReason('email_verify_required', None)

is_email_verify_required()
Check if the union tag is email_verify_required.

Return type bool

is_login_required()
Check if the union tag is login_required.

Return type bool

is_other()
Check if the union tag is other.

Return type bool

is_owner_only()
Check if the union tag is owner_only.

Return type bool

is_password_required()
Check if the union tag is password_required.

Return type bool

is_team_only()
Check if the union tag is team_only.

Return type bool

login_required = SharedLinkAccessFailureReason('login_required', None)

other = SharedLinkAccessFailureReason('other', None)

owner_only = SharedLinkAccessFailureReason('owner_only', None)

password_required = SharedLinkAccessFailureReason('password_required', None)

team_only = SharedLinkAccessFailureReason('team_only', None)

class dropbox.sharing.**SharedLinkError**(*tag*, *value=None*)

Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables

- ***shared_link_not_found*** – The shared link wasn't found
- ***shared_link_access_denied*** – The caller is not allowed to access this shared link

is_other()

Check if the union tag is other.

Return type bool

is_shared_link_access_denied()

Check if the union tag is shared_link_access_denied.

Return type bool

is_shared_link_not_found()

Check if the union tag is shared_link_not_found.

Return type bool

other = SharedLinkError('other', None)

shared_link_access_denied = SharedLinkError('shared_link_access_denied', None)

shared_link_not_found = SharedLinkError('shared_link_not_found', None)

class dropbox.sharing.**SharedLinkMetadata**(*url=None*, *name=None*, *link_permissions=None*, *id=None*, *expires=None*, *path_lower=None*, *team_member_info=None*, *content_owner_team_info=None*)

Bases: object

The metadata of a shared link

Variables

- ***url*** – URL of the shared link.
- ***id*** – A unique identifier for the linked file.
- ***name*** – The linked file name (including extension). This never contains a slash.
- ***expires*** – Expiration time, if set. By default the link won't expire.
- ***path_lower*** – The lowercased full path in the user's Dropbox. This always starts with a slash. This field will only be present only if the linked file is in the authenticated user's dropbox.
- ***link_permissions*** – The link's access permissions.

- ***team_member_info*** – The team membership information of the link's owner. This field will only be present if the link's owner is a team member.
- ***content_owner_team_info*** – The team information of the content's owner. This field will only be present if the content's owner is a team member and the content's owner team is different from the link's owner team.

__init__(*url=None*, *name=None*, *link_permissions=None*, *id=None*, *expires=None*, *path_lower=None*, *team_member_info=None*, *content_owner_team_info=None*)

content_owner_team_info

The team information of the content's owner. This field will only be present if the content's owner is a team member and the content's owner team is different from the link's owner team.

Return type users.Team_validator

expires

Expiration time, if set. By default the link won't expire.

Return type datetime.datetime

id

A unique identifier for the linked file.

Return type str

link_permissions

The link's access permissions.

Return type *LinkPermissions*

name

The linked file name (including extension). This never contains a slash.

Return type str

path_lower

The lowercased full path in the user's Dropbox. This always starts with a slash. This field will only be present only if the linked file is in the authenticated user's dropbox.

Return type str

team_member_info

The team membership information of the link's owner. This field will only be present if the link's owner is a team member.

Return type *TeamMemberInfo*

url

URL of the shared link.

Return type str

class dropbox.sharing.**SharedLinkPolicy**(*tag*, *value=None*)

Bases: dropbox.stone_base.Union

Policy governing who can view shared links.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***anyone*** – Links can be shared with anyone.
- ***members*** – Links can only be shared among members of the shared folder.

anyone = SharedLinkPolicy('anyone', None)

is_anyone()
Check if the union tag is `anyone`.

Return type bool

is_members()
Check if the union tag is `members`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

members = SharedLinkPolicy('members', None)

other = SharedLinkPolicy('other', None)

class `dropbox.sharing.`**SharedLinkSettings**(*requested_visibility=None, link_password=None, expires=None*)

Bases: `object`

Variables

- ***requested_visibility*** – The requested access for this shared link.
- ***link_password*** – If `requested_visibility` is `RequestedVisibility.password` this is needed to specify the password to access the link.
- ***expires*** – Expiration time of the shared link. By default the link won't expire.

__init__(*requested_visibility=None, link_password=None, expires=None*)

expires
Expiration time of the shared link. By default the link won't expire.

Return type datetime.datetime

link_password
If `requested_visibility` is `RequestedVisibility.password` this is needed to specify the password to access the link.

Return type str

requested_visibility
The requested access for this shared link.

Return type *RequestedVisibility*

class `dropbox.sharing.`**SharedLinkSettingsError**(*tag, value=None*)

Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***invalid_settings*** – The given settings are invalid (for example, all attributes of the *SharedLinkSettings* are empty, the requested visibility is `RequestedVisibility.password` but the `SharedLinkSettings.link_password` is missing, `SharedLinkSettings.expires` is set to the past, etc.)

- ***not_authorized*** – User is not allowed to modify the settings of this link. Note that basic users can only set `RequestedVisibility.public` as the `SharedLinkSettings.requested_visibility` and cannot set `SharedLinkSettings.expires`

invalid_settings = SharedLinkSettingsError('invalid_settings', None)

is_invalid_settings()
Check if the union tag is `invalid_settings`.

Return type bool

is_not_authorized()
Check if the union tag is `not_authorized`.

Return type bool

not_authorized = SharedLinkSettingsError('not_authorized', None)

class dropbox.sharing.**SharingFileAccessError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

User could not access this file.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***no_permission*** – Current user does not have sufficient privileges to perform the desired action.
- ***invalid_file*** – File specified was not found.
- ***is_folder*** – A folder can't be shared this way. Use folder sharing or a shared link instead.
- ***inside_public_folder*** – A file inside a public folder can't be shared this way. Use a public link instead.
- ***inside_osx_package*** – A Mac OS X package can't be shared this way. Use a shared link instead.

inside_osx_package = SharingFileAccessError('inside_osx_package', None)

inside_public_folder = SharingFileAccessError('inside_public_folder', None)

invalid_file = SharingFileAccessError('invalid_file', None)

is_folder = SharingFileAccessError('is_folder', None)

is_inside_osx_package()
Check if the union tag is `inside_osx_package`.

Return type bool

is_inside_public_folder()
Check if the union tag is `inside_public_folder`.

Return type bool

is_invalid_file()
Check if the union tag is `invalid_file`.

Return type bool

is_is_folder()
Check if the union tag is `is_folder`.

Return type bool

is_no_permission()
Check if the union tag is `no_permission`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

no_permission = SharingFileAccessError('no_permission', None)

other = SharingFileAccessError('other', None)

class dropbox.sharing.**SharingUserError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

User account had a problem preventing this action.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***email_unverified*** – The current user must verify the account e-mail address before performing this action.

email_unverified = SharingUserError('email_unverified', None)

is_email_unverified()
Check if the union tag is `email_unverified`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

other = SharingUserError('other', None)

class dropbox.sharing.**TeamMemberInfo**(*team_info=None*, *display_name=None*, *member_id=None*)
Bases: `object`

Information about a team member.

Variables

- ***team_info*** – Information about the member's team
- ***display_name*** – The display name of the user.
- ***member_id*** – ID of user as a member of a team. This field will only be present if the member is in the same team as current user.

__init__(*team_info=None*, *display_name=None*, *member_id=None*)

display_name
The display name of the user.

Return type str

member_id
ID of user as a member of a team. This field will only be present if the member is in the same team as current user.

Return type str

team_info
Information about the member's team

Return type users.Team_validator

class dropbox.sharing.**TransferFolderArg**(*shared_folder_id=None*, *to_dropbox_id=None*)
Bases: object

Variables

- ***shared_folder_id*** – The ID for the shared folder.
- ***to_dropbox_id*** – A account or team member ID to transfer ownership to.

__init__(*shared_folder_id=None*, *to_dropbox_id=None*)

shared_folder_id
The ID for the shared folder.

Return type str

to_dropbox_id
A account or team member ID to transfer ownership to.

Return type str

class dropbox.sharing.**TransferFolderError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***invalid_dropbox_id*** – `TransferFolderArg.to_dropbox_id` is invalid.
- ***new_owner_not_a_member*** – The new designated owner is not currently a member of the shared folder.
- ***new_owner_unmounted*** – The new designated owner has not added the folder to their Dropbox.
- ***new_owner_email_unverified*** – The new designated owner's e-mail address is unverified.
- ***team_folder*** – This action cannot be performed on a team shared folder.
- ***no_permission*** – The current user does not have permission to perform this action.

classmethod access_error(*val*)
Create an instance of this class set to the `access_error` tag with value `val`.

Parameters **val** (SharedFolderAccessError) –

Return type *TransferFolderError*

get_access_error()
Only call this if *is_access_error()* is true.

Return type *SharedFolderAccessError*

invalid_dropbox_id = TransferFolderError('invalid_dropbox_id', None)

is_access_error()
Check if the union tag is `access_error`.

Return type bool

is_invalid_dropbox_id()
Check if the union tag is `invalid_dropbox_id`.

Return type bool

is_new_owner_email_unverified()
Check if the union tag is `new_owner_email_unverified`.

Return type bool

is_new_owner_not_a_member()
Check if the union tag is `new_owner_not_a_member`.

Return type bool

is_new_owner_unmounted()
Check if the union tag is `new_owner_unmounted`.

Return type bool

is_no_permission()
Check if the union tag is `no_permission`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_team_folder()
Check if the union tag is `team_folder`.

Return type bool

new_owner_email_unverified = TransferFolderError('new_owner_email_unverified', None)

new_owner_not_a_member = TransferFolderError('new_owner_not_a_member', None)

new_owner_unmounted = TransferFolderError('new_owner_unmounted', None)

no_permission = TransferFolderError('no_permission', None)

other = TransferFolderError('other', None)

team_folder = TransferFolderError('team_folder', None)

class dropbox.sharing.**UnmountFolderArg**(*shared_folder_id=None*)
Bases: `object`

Variables ***shared_folder_id*** – The ID for the shared folder.

__init__(*shared_folder_id=None*)

shared_folder_id
The ID for the shared folder.

Return type str

class dropbox.sharing.**UnmountFolderError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***no_permission*** – The current user does not have permission to perform this action.

- ***not_unmountable*** – The shared folder can't be unmounted. One example where this can occur is when the shared folder's parent folder is also a shared folder that resides in the current user's Dropbox.

classmethod access_error(*val*)
Create an instance of this class set to the `access_error` tag with value `val`.

Parameters **val** (`SharedFolderAccessError`) –

Return type *UnmountFolderError*

get_access_error()
Only call this if *`is_access_error()`* is true.

Return type *SharedFolderAccessError*

is_access_error()
Check if the union tag is `access_error`.

Return type bool

is_no_permission()
Check if the union tag is `no_permission`.

Return type bool

is_not_unmountable()
Check if the union tag is `not_unmountable`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

no_permission = UnmountFolderError('no_permission', None)

not_unmountable = UnmountFolderError('not_unmountable', None)

other = UnmountFolderError('other', None)

class dropbox.sharing.**UnshareFileArg**(*file=None*)
Bases: `object`

Arguments for *`dropbox.dropbox.Dropbox.sharing_unshare_file()`*.

Variables ***file*** – The file to unshare.

__init__(*file=None*)

file
The file to unshare.

Return type str

class dropbox.sharing.**UnshareFileError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Error result for *`dropbox.dropbox.Dropbox.sharing_unshare_file()`*.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

classmethod access_error(*val*)
Create an instance of this class set to the `access_error` tag with value `val`.

Parameters **val** (SharingFileAccessError) –

Return type *UnshareFileError*

get_access_error()
Only call this if *is_access_error()* is true.

Return type *SharingFileAccessError*

get_user_error()
Only call this if *is_user_error()* is true.

Return type *SharingUserError*

is_access_error()
Check if the union tag is access_error.

Return type bool

is_other()
Check if the union tag is other.

Return type bool

is_user_error()
Check if the union tag is user_error.

Return type bool

other = UnshareFileError('other', None)

classmethod user_error(*val*)
Create an instance of this class set to the user_error tag with value val.

Parameters **val** (SharingUserError) –

Return type *UnshareFileError*

class dropbox.sharing.**UnshareFolderArg**(*shared_folder_id=None*, *leave_a_copy=None*)
Bases: object

Variables

- ***shared_folder_id*** – The ID for the shared folder.
- ***leave_a_copy*** – If true, members of this shared folder will get a copy of this folder after it's unshared. Otherwise, it will be removed from their Dropbox. The current user, who is an owner, will always retain their copy.

__init__(*shared_folder_id=None*, *leave_a_copy=None*)

leave_a_copy
If true, members of this shared folder will get a copy of this folder after it's unshared. Otherwise, it will be removed from their Dropbox. The current user, who is an owner, will always retain their copy.

Return type bool

shared_folder_id
The ID for the shared folder.

Return type str

class dropbox.sharing.**UnshareFolderError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables

- ***team_folder*** – This action cannot be performed on a team shared folder.
- ***no_permission*** – The current user does not have permission to perform this action.

classmethod access_error(*val*)

Create an instance of this class set to the `access_error` tag with value `val`.

Parameters **val** (`SharedFolderAccessError`) –

Return type *UnshareFolderError*

get_access_error()

Only call this if *`is_access_error()`* is true.

Return type *SharedFolderAccessError*

is_access_error()

Check if the union tag is `access_error`.

Return type bool

is_no_permission()

Check if the union tag is `no_permission`.

Return type bool

is_other()

Check if the union tag is `other`.

Return type bool

is_team_folder()

Check if the union tag is `team_folder`.

Return type bool

no_permission = UnshareFolderError('no_permission', None)

other = UnshareFolderError('other', None)

team_folder = UnshareFolderError('team_folder', None)

class dropbox.sharing.**UpdateFolderMemberArg**(*shared_folder_id=None*, *member=None*, *access_level=None*)

Bases: `object`

Variables

- ***shared_folder_id*** – The ID for the shared folder.
- ***member*** – The member of the shared folder to update. Only the `MemberSelector.dropbox_id` may be set at this time.
- ***access_level*** – The new access level for `member`. `AccessLevel.owner` is disallowed.

__init__(*shared_folder_id=None*, *member=None*, *access_level=None*)

access_level

The new access level for `member`. `AccessLevel.owner` is disallowed.

Return type *AccessLevel*

member

The member of the shared folder to update. Only the `MemberSelector.dropbox_id` may be set at this time.

Return type *MemberSelector*

shared_folder_id
The ID for the shared folder.

Return type str

class dropbox.sharing.**UpdateFolderMemberError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables

- ***no_explicit_access*** (AddFolderMemberError) – If updating the access type required the member to be added to the shared folder and there was an error when adding the member.
- ***insufficient_plan*** – The current user's account doesn't support this action. An example of this is when downgrading a member from editor to viewer. This action can only be performed by users that have upgraded to a Pro or Business plan.
- ***no_permission*** – The current user does not have permission to perform this action.

classmethod access_error(*val*)
Create an instance of this class set to the access_error tag with value val.

Parameters **val** (SharedFolderAccessError) –

Return type *UpdateFolderMemberError*

get_access_error()
Only call this if *is_access_error()* is true.

Return type *SharedFolderAccessError*

get_member_error()
Only call this if *is_member_error()* is true.

Return type *SharedFolderMemberError*

get_no_explicit_access()
If updating the access type required the member to be added to the shared folder and there was an error when adding the member.

Only call this if *is_no_explicit_access()* is true.

Return type *AddFolderMemberError*

insufficient_plan = UpdateFolderMemberError('insufficient_plan', None)

is_access_error()
Check if the union tag is access_error.

Return type bool

is_insufficient_plan()
Check if the union tag is insufficient_plan.

Return type bool

is_member_error()
Check if the union tag is member_error.

Return type bool

is_no_explicit_access()
Check if the union tag is `no_explicit_access`.

Return type bool

is_no_permission()
Check if the union tag is `no_permission`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

classmethod member_error(*val*)
Create an instance of this class set to the `member_error` tag with value `val`.

Parameters **val** (`SharedFolderMemberError`) –

Return type *UpdateFolderMemberError*

classmethod no_explicit_access(*val*)
Create an instance of this class set to the `no_explicit_access` tag with value `val`.

Parameters **val** (`AddFolderMemberError`) –

Return type *UpdateFolderMemberError*

no_permission = UpdateFolderMemberError('no_permission', None)

other = UpdateFolderMemberError('other', None)

class `dropbox.sharing.`**UpdateFolderPolicyArg**(*shared_folder_id=None*, *member_policy=None*, *acl_update_policy=None*, *shared_link_policy=None*)

Bases: `object`

If any of the policy's are unset, then they retain their current setting.

Variables

- ***shared_folder_id*** – The ID for the shared folder.
- ***member_policy*** – Who can be a member of this shared folder. Only applicable if the current user is on a team.
- ***acl_update_policy*** – Who can add and remove members of this shared folder.
- ***shared_link_policy*** – The policy to apply to shared links created for content inside this shared folder. The current user must be on a team to set this policy to `SharedLinkPolicy.members`.

__init__(*shared_folder_id=None*, *member_policy=None*, *acl_update_policy=None*, *shared_link_policy=None*)

acl_update_policy
Who can add and remove members of this shared folder.

Return type *AclUpdatePolicy*

member_policy
Who can be a member of this shared folder. Only applicable if the current user is on a team.

Return type *MemberPolicy*

shared_folder_id
The ID for the shared folder.

Return type str

shared_link_policy
The policy to apply to shared links created for content inside this shared folder. The current user must be on a team to set this policy to `SharedLinkPolicy.members`.

Return type *SharedLinkPolicy*

class `dropbox.sharing.`**UpdateFolderPolicyError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***not_on_team*** – `UpdateFolderPolicyArg.member_policy` was set even though user is not on a team.
- ***team_policy_disallows_member_policy*** – Team policy is more restrictive than `ShareFolderArg.member_policy`.
- ***disallowed_shared_link_policy*** – The current account is not allowed to select the specified `ShareFolderArg.shared_link_policy`.
- ***no_permission*** – The current user does not have permission to perform this action.

classmethod access_error(*val*)
Create an instance of this class set to the `access_error` tag with value `val`.

Parameters **val** (`SharedFolderAccessError`) –

Return type *UpdateFolderPolicyError*

disallowed_shared_link_policy = UpdateFolderPolicyError('disallowed_shared_link_policy', None)

get_access_error()
Only call this if *`is_access_error()`* is true.

Return type *SharedFolderAccessError*

is_access_error()
Check if the union tag is `access_error`.

Return type bool

is_disallowed_shared_link_policy()
Check if the union tag is `disallowed_shared_link_policy`.

Return type bool

is_no_permission()
Check if the union tag is `no_permission`.

Return type bool

is_not_on_team()
Check if the union tag is `not_on_team`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_team_policy_disallows_member_policy()
Check if the union tag is `team_policy_disallows_member_policy`.

Return type bool

no_permission = UpdateFolderPolicyError('no_permission', None)

not_on_team = UpdateFolderPolicyError('not_on_team', None)

other = UpdateFolderPolicyError('other', None)

team_policy_disallows_member_policy = UpdateFolderPolicyError('team_policy_disallows_member_policy', N

class dropbox.sharing.**UserInfo**(*account_id=None*, *same_team=None*, *team_member_id=None*)
Bases: `object`

Basic information about a user. Use `dropbox.dropbox.Dropbox.sharing_users_account()` and `dropbox.dropbox.Dropbox.sharing_users_account_batch()` to obtain more detailed information.

Variables

- ***account_id*** – The account ID of the user.
- ***same_team*** – If the user is in the same team as current user.
- ***team_member_id*** – The team member ID of the shared folder member. Only present if `same_team` is true.

__init__(*account_id=None*, *same_team=None*, *team_member_id=None*)

account_id
The account ID of the user.

Return type str

same_team
If the user is in the same team as current user.

Return type bool

team_member_id
The team member ID of the shared folder member. Only present if `same_team` is true.

Return type str

class dropbox.sharing.**UserMembershipInfo**(*access_type=None*, *user=None*, *permissions=None*, *initials=None*, *is_inherited=None*)
Bases: *dropbox.sharing.MembershipInfo*

The information about a user member of the shared content.

Variables ***user*** – The account information for the membership user.

__init__(*access_type=None*, *user=None*, *permissions=None*, *initials=None*, *is_inherited=None*)

user
The account information for the membership user.

Return type *UserInfo*

class dropbox.sharing.**Visibility**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Who can access a shared link. The most open visibility is `public`. The default depends on many aspects, such as team and user preferences and shared folder settings.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***public*** – Anyone who has received the link can access it. No login required.
- ***team_only*** – Only members of the same team can access the link. Login is required.
- ***password*** – A link-specific password is required to access the link. Login is not required.
- ***team_and_password*** – Only members of the same team who have the link-specific password can access the link.
- ***shared_folder_only*** – Only members of the shared folder containing the linked file can access the link. Login is required.

is_other()
Check if the union tag is `other`.

Return type bool

is_password()
Check if the union tag is `password`.

Return type bool

is_public()
Check if the union tag is `public`.

Return type bool

is_shared_folder_only()
Check if the union tag is `shared_folder_only`.

Return type bool

is_team_and_password()
Check if the union tag is `team_and_password`.

Return type bool

is_team_only()
Check if the union tag is `team_only`.

Return type bool

other = Visibility('other', None)

password = Visibility('password', None)

public = Visibility('public', None)

shared_folder_only = Visibility('shared_folder_only', None)

team_and_password = Visibility('team_and_password', None)

team_only = Visibility('team_only', None)

2.6 dropbox.team – Team

class dropbox.team.**ActiveWebSession**(*session_id=None*, *user_agent=None*, *os=None*, *browser=None*, *ip_address=None*, *country=None*, *created=None*, *updated=None*)

Bases: *dropbox.team.DeviceSession*

Information on active web sessions

Variables

- ***user_agent*** – Information on the hosting device
- ***os*** – Information on the hosting operating system
- ***browser*** – Information on the browser used for this web session

__init__(*session_id=None*, *user_agent=None*, *os=None*, *browser=None*, *ip_address=None*, *country=None*, *created=None*, *updated=None*)

browser

Information on the browser used for this web session

Return type str

os

Information on the hosting operating system

Return type str

user_agent

Information on the hosting device

Return type str

class dropbox.team.**AddPropertyTemplateArg**(*name=None*, *description=None*, *fields=None*)

Bases: dropbox.properties.PropertyGroupTemplate

Arguments for adding property templates.

__init__(*name=None*, *description=None*, *fields=None*)

class dropbox.team.**AddPropertyTemplateResult**(*template_id=None*)

Bases: object

Variables ***template_id*** – An identifier for property template added by dropbox.dropbox.Dropbox.team_properties_template_add().

__init__(*template_id=None*)

template_id

An identifier for property template added by dropbox.dropbox.Dropbox.team_properties_template_add(

Return type str

class dropbox.team.**AdminTier**(*tag*, *value=None*)

Bases: dropbox.stone_base.Union

Describes which team-related admin permissions a user has.

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables

- ***team_admin*** – User is an administrator of the team - has all permissions.

- ***user_management_admin*** – User can do most user provisioning, de-provisioning and management.
- ***support_admin*** – User can do a limited set of common support tasks for existing users.
- ***member_only*** – User is not an admin of the team.

is_member_only()
Check if the union tag is `member_only`.

Return type bool

is_support_admin()
Check if the union tag is `support_admin`.

Return type bool

is_team_admin()
Check if the union tag is `team_admin`.

Return type bool

is_user_management_admin()
Check if the union tag is `user_management_admin`.

Return type bool

member_only = AdminTier('member_only', None)

support_admin = AdminTier('support_admin', None)

team_admin = AdminTier('team_admin', None)

user_management_admin = AdminTier('user_management_admin', None)

class dropbox.team.**ApiApp**(*app_id=None*, *app_name=None*, *is_app_folder=None*, *publisher=None*, *publisher_url=None*, *linked=None*)

Bases: `object`

Information on linked third party applications

Variables

- ***app_id*** – The application unique id
- ***app_name*** – The application name
- ***publisher*** – The application publisher name
- ***publisher_url*** – The publisher's URL
- ***linked*** – The time this application was linked
- ***is_app_folder*** – Whether the linked application uses a dedicated folder

__init__(*app_id=None*, *app_name=None*, *is_app_folder=None*, *publisher=None*, *publisher_url=None*, *linked=None*)

app_id
The application unique id

Return type str

app_name
The application name

Return type str

is_app_folder
Whether the linked application uses a dedicated folder

Return type bool

linked
The time this application was linked

Return type datetime.datetime

publisher
The application publisher name

Return type str

publisher_url
The publisher's URL

Return type str

class dropbox.team.**BaseDfbReport**(*start_date=None*)
Bases: object

Base report structure.

Variables ***start_date*** – First date present in the results as 'YYYY-MM-DD' or None.

__init__(*start_date=None*)

start_date
First date present in the results as 'YYYY-MM-DD' or None.

Return type str

class dropbox.team.**DateRange**(*start_date=None*, *end_date=None*)
Bases: object

Input arguments that can be provided for most reports.

Variables

- ***start_date*** – Optional starting date (inclusive)
- ***end_date*** – Optional ending date (exclusive)

__init__(*start_date=None*, *end_date=None*)

end_date
Optional ending date (exclusive)

Return type datetime.datetime

start_date
Optional starting date (inclusive)

Return type datetime.datetime

class dropbox.team.**DateRangeError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

Errors that can originate from problems in input arguments to reports.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

is_other()
Check if the union tag is `other`.

Return type bool

other = DateRangeError('other', None)

class dropbox.team.**DesktopClientSession**(*session_id=None*, *host_name=None*, *client_type=None*, *client_version=None*, *platform=None*, *is_delete_on_unlink_supported=None*, *ip_address=None*, *country=None*, *created=None*, *updated=None*)

Bases: *dropbox.team.DeviceSession*

Information about linked Dropbox desktop client sessions

Variables

- ***host_name*** – Name of the hosting desktop
- ***client_type*** – The Dropbox desktop client type
- ***client_version*** – The Dropbox client version
- ***platform*** – Information on the hosting platform
- ***is_delete_on_unlink_supported*** – Whether it's possible to delete all of the account files upon unlinking

__init__(*session_id=None*, *host_name=None*, *client_type=None*, *client_version=None*, *platform=None*, *is_delete_on_unlink_supported=None*, *ip_address=None*, *country=None*, *created=None*, *updated=None*)

client_type

The Dropbox desktop client type

Return type *DesktopPlatform*

client_version

The Dropbox client version

Return type str

host_name

Name of the hosting desktop

Return type str

is_delete_on_unlink_supported

Whether it's possible to delete all of the account files upon unlinking

Return type bool

platform

Information on the hosting platform

Return type str

class dropbox.team.**DesktopPlatform**(*tag*, *value=None*)

Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***windows*** – Official Windows Dropbox desktop client
- ***mac*** – Official Mac Dropbox desktop client

- ***linux*** – Official Linux Dropbox desktop client

is_linux()
Check if the union tag is `linux`.

Return type bool

is_mac()
Check if the union tag is `mac`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_windows()
Check if the union tag is `windows`.

Return type bool

linux = DesktopPlatform('linux', None)

mac = DesktopPlatform('mac', None)

other = DesktopPlatform('other', None)

windows = DesktopPlatform('windows', None)

class dropbox.team.**DeviceSession**(*session_id=None*, *ip_address=None*, *country=None*, *created=None*, *updated=None*)

Bases: `object`

Variables

- ***session_id*** – The session id
- ***ip_address*** – The IP address of the last activity from this session
- ***country*** – The country from which the last activity from this session was made
- ***created*** – The time this session was created
- ***updated*** – The time of the last activity from this session

__init__(*session_id=None*, *ip_address=None*, *country=None*, *created=None*, *updated=None*)

country
The country from which the last activity from this session was made

Return type str

created
The time this session was created

Return type datetime.datetime

ip_address
The IP address of the last activity from this session

Return type str

session_id
The session id

Return type str

updated
The time of the last activity from this session

Return type datetime.datetime

class `dropbox.team.`**`DeviceSessionArg`**(*session_id=None*, *team_member_id=None*)
Bases: `object`

Variables

- ***session_id*** – The session id
- ***team_member_id*** – The unique id of the member owning the device

`__init__`(*session_id=None*, *team_member_id=None*)

session_id
The session id

Return type str

team_member_id
The unique id of the member owning the device

Return type str

class `dropbox.team.`**`DevicesActive`**(*windows=None*, *macos=None*, *linux=None*, *ios=None*, *android=None*, *other=None*, *total=None*)
Bases: `object`

Each of the items is an array of values, one value per day. The value is the number of devices active within a time window, ending with that day. If there is no data for a day, then the value will be None.

Variables

- ***windows*** – Array of number of linked windows (desktop) clients with activity.
- ***macos*** – Array of number of linked mac (desktop) clients with activity.
- ***linux*** – Array of number of linked linus (desktop) clients with activity.
- ***ios*** – Array of number of linked ios devices with activity.
- ***android*** – Array of number of linked android devices with activity.
- ***other*** – Array of number of other linked devices (blackberry, windows phone, etc) with activity.
- ***total*** – Array of total number of linked clients with activity.

`__init__`(*windows=None*, *macos=None*, *linux=None*, *ios=None*, *android=None*, *other=None*, *total=None*)

android
Array of number of linked android devices with activity.

Return type list of [Optional[long]]

ios
Array of number of linked ios devices with activity.

Return type list of [Optional[long]]

linux
Array of number of linked linus (desktop) clients with activity.

Return type list of [Optional[long]]

macos
Array of number of linked mac (desktop) clients with activity.

Return type list of [Optional[long]]

other
Array of number of other linked devices (blackberry, windows phone, etc) with activity.

Return type list of [Optional[long]]

total
Array of total number of linked clients with activity.

Return type list of [Optional[long]]

windows
Array of number of linked windows (desktop) clients with activity.

Return type list of [Optional[long]]

class `dropbox.team.`**`GetActivityReport`**(*start_date=None*, *adds=None*, *edits=None*, *deletes=None*, *active_users_28_day=None*, *active_users_7_day=None*, *active_users_1_day=None*, *active_shared_folders_28_day=None*, *active_shared_folders_7_day=None*, *active_shared_folders_1_day=None*, *shared_links_created=None*, *shared_links_viewed_by_team=None*, *shared_links_viewed_by_outside_user=None*, *shared_links_viewed_by_not_logged_in=None*, *shared_links_viewed_total=None*)

Bases: `dropbox.team.BaseDfbReport`

Activity Report Result. Each of the items in the storage report is an array of values, one value per day. If there is no data for a day, then the value will be None.

Variables

- ***adds*** – Array of total number of adds by team members.
- ***edits*** – Array of number of edits by team members. If the same user edits the same file multiple times this is counted as a single edit.
- ***deletes*** – Array of total number of deletes by team members.
- ***active_users_28_day*** – Array of the number of users who have been active in the last 28 days.
- ***active_users_7_day*** – Array of the number of users who have been active in the last week.
- ***active_users_1_day*** – Array of the number of users who have been active in the last day.
- ***active_shared_folders_28_day*** – Array of the number of shared folders with some activity in the last 28 days.
- ***active_shared_folders_7_day*** – Array of the number of shared folders with some activity in the last week.
- ***active_shared_folders_1_day*** – Array of the number of shared folders with some activity in the last day.
- ***shared_links_created*** – Array of the number of shared links created.

- ***shared_links_viewed_by_team*** – Array of the number of views by team users to shared links created by the team.
- ***shared_links_viewed_by_outside_user*** – Array of the number of views by users outside of the team to shared links created by the team.
- ***shared_links_viewed_by_not_logged_in*** – Array of the number of views by non-logged-in users to shared links created by the team.
- ***shared_links_viewed_total*** – Array of the total number of views to shared links created by the team.

__init__(*start_date=None*, *adds=None*, *edits=None*, *deletes=None*, *active_users_28_day=None*, *active_users_7_day=None*, *active_users_1_day=None*, *active_shared_folders_28_day=None*, *active_shared_folders_7_day=None*, *active_shared_folders_1_day=None*, *shared_links_created=None*, *shared_links_viewed_by_team=None*, *shared_links_viewed_by_outside_user=None*, *shared_links_viewed_by_not_logged_in=None*, *shared_links_viewed_total=None*)

active_shared_folders_1_day

Array of the number of shared folders with some activity in the last day.

Return type list of [Optional[long]]

active_shared_folders_28_day

Array of the number of shared folders with some activity in the last 28 days.

Return type list of [Optional[long]]

active_shared_folders_7_day

Array of the number of shared folders with some activity in the last week.

Return type list of [Optional[long]]

active_users_1_day

Array of the number of users who have been active in the last day.

Return type list of [Optional[long]]

active_users_28_day

Array of the number of users who have been active in the last 28 days.

Return type list of [Optional[long]]

active_users_7_day

Array of the number of users who have been active in the last week.

Return type list of [Optional[long]]

adds

Array of total number of adds by team members.

Return type list of [Optional[long]]

deletes

Array of total number of deletes by team members.

Return type list of [Optional[long]]

edits

Array of number of edits by team members. If the same user edits the same file multiple times this is counted as a single edit.

Return type list of [Optional[long]]

shared_links_created
Array of the number of shared links created.

Return type list of [Optional[long]]

shared_links_viewed_by_not_logged_in
Array of the number of views by non-logged-in users to shared links created by the team.

Return type list of [Optional[long]]

shared_links_viewed_by_outside_user
Array of the number of views by users outside of the team to shared links created by the team.

Return type list of [Optional[long]]

shared_links_viewed_by_team
Array of the number of views by team users to shared links created by the team.

Return type list of [Optional[long]]

shared_links_viewed_total
Array of the total number of views to shared links created by the team.

Return type list of [Optional[long]]

class dropbox.team.**GetDevicesReport**(*start_date=None*, *active_1_day=None*, *active_7_day=None*, *active_28_day=None*)
Bases: *dropbox.team.BaseDfbReport*

Devices Report Result. Contains subsections for different time ranges of activity. Each of the items in each subsection of the storage report is an array of values, one value per day. If there is no data for a day, then the value will be None.

Variables

- ***active_1_day*** – Report of the number of devices active in the last day.
- ***active_7_day*** – Report of the number of devices active in the last 7 days.
- ***active_28_day*** – Report of the number of devices active in the last 28 days.

__init__(*start_date=None*, *active_1_day=None*, *active_7_day=None*, *active_28_day=None*)

active_1_day
Report of the number of devices active in the last day.

Return type *DevicesActive*

active_28_day
Report of the number of devices active in the last 28 days.

Return type *DevicesActive*

active_7_day
Report of the number of devices active in the last 7 days.

Return type *DevicesActive*

class dropbox.team.**GetMembershipReport**(*start_date=None*, *team_size=None*, *pending_invites=None*, *members_joined=None*, *suspended_members=None*, *licenses=None*)
Bases: *dropbox.team.BaseDfbReport*

Membership Report Result. Each of the items in the storage report is an array of values, one value per day. If there is no data for a day, then the value will be None.

Variables

- ***team_size*** – Team size, for each day.
- ***pending_invites*** – The number of pending invites to the team, for each day.
- ***members_joined*** – The number of members that joined the team, for each day.
- ***suspended_members*** – The number of suspended team members, for each day.
- ***licenses*** – The total number of licenses the team has, for each day.

__init__(*start_date=None*, *team_size=None*, *pending_invites=None*, *members_joined=None*, *suspended_members=None*, *licenses=None*)

licenses

The total number of licenses the team has, for each day.

Return type list of [Optional[long]]

members_joined

The number of members that joined the team, for each day.

Return type list of [Optional[long]]

pending_invites

The number of pending invites to the team, for each day.

Return type list of [Optional[long]]

suspended_members

The number of suspended team members, for each day.

Return type list of [Optional[long]]

team_size

Team size, for each day.

Return type list of [Optional[long]]

class dropbox.team.**GetStorageReport**(*start_date=None*, *total_usage=None*, *shared_usage=None*, *unshared_usage=None*, *shared_folders=None*, *member_storage_map=None*)

Bases: *dropbox.team.BaseDfbReport*

Storage Report Result. Each of the items in the storage report is an array of values, one value per day. If there is no data for a day, then the value will be None.

Variables

- ***total_usage*** – Sum of the shared, unshared, and datastore usages, for each day.
- ***shared_usage*** – Array of the combined size (bytes) of team members' shared folders, for each day.
- ***unshared_usage*** – Array of the combined size (bytes) of team members' root namespaces, for each day.
- ***shared_folders*** – Array of the number of shared folders owned by team members, for each day.
- ***member_storage_map*** – Array of storage summaries of team members' account sizes. Each storage summary is an array of key, value pairs, where each pair describes a storage bucket. The key indicates the upper bound of the bucket and the value is the number of users in that bucket. There is one such summary per day. If there is no data for a day, the storage summary will be empty.

__init__(*start_date=None*, *total_usage=None*, *shared_usage=None*, *unshared_usage=None*, *shared_folders=None*, *member_storage_map=None*)

member_storage_map
Array of storage summaries of team members' account sizes. Each storage summary is an array of key, value pairs, where each pair describes a storage bucket. The key indicates the upper bound of the bucket and the value is the number of users in that bucket. There is one such summary per day. If there is no data for a day, the storage summary will be empty.

Return type list of [list of [StorageBucket]]

shared_folders
Array of the number of shared folders owned by team members, for each day.

Return type list of [Optional[long]]

shared_usage
Array of the combined size (bytes) of team members' shared folders, for each day.

Return type list of [Optional[long]]

total_usage
Sum of the shared, unshared, and datastore usages, for each day.

Return type list of [Optional[long]]

unshared_usage
Array of the combined size (bytes) of team members' root namespaces, for each day.

Return type list of [Optional[long]]

class dropbox.team.**GroupAccessType**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

Role of a user in group.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***member*** – User is a member of the group, but has no special permissions.
- ***owner*** – User can rename the group, and add/remove members.

is_member()
Check if the union tag is `member`.

Return type bool

is_owner()
Check if the union tag is `owner`.

Return type bool

member = GroupAccessType('member', None)

owner = GroupAccessType('owner', None)

class dropbox.team.**GroupCreateArg**(*group_name=None*, *group_external_id=None*, *group_management_type=None*)
Bases: object

Variables

- ***group_name*** – Group name.
- ***group_external_id*** – The creator of a team can associate an arbitrary external ID to the group.

- ***group_management_type*** – Whether the team can be managed by selected users, or only by team admins

__init__(*group_name=None*, *group_external_id=None*, *group_management_type=None*)

group_external_id
The creator of a team can associate an arbitrary external ID to the group.

Return type str

group_management_type
Whether the team can be managed by selected users, or only by team admins

Return type team_common.GroupManagementType_validator

group_name
Group name.

Return type str

class dropbox.team.**GroupCreateError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***group_name_already_used*** – There is already an existing group with the requested name.
- ***group_name_invalid*** – Group name is empty or has invalid characters.
- ***external_id_already_in_use*** – The new external ID is already being used by another group.

external_id_already_in_use = GroupCreateError('external_id_already_in_use', None)

group_name_already_used = GroupCreateError('group_name_already_used', None)

group_name_invalid = GroupCreateError('group_name_invalid', None)

is_external_id_already_in_use()
Check if the union tag is `external_id_already_in_use`.

Return type bool

is_group_name_already_used()
Check if the union tag is `group_name_already_used`.

Return type bool

is_group_name_invalid()
Check if the union tag is `group_name_invalid`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

other = GroupCreateError('other', None)

class dropbox.team.**GroupDeleteError**(*tag*, *value=None*)
Bases: *dropbox.team.GroupSelectorError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***group_already_deleted*** – This group has already been deleted.

group_already_deleted = GroupDeleteError('group_already_deleted', None)

is_group_already_deleted()
Check if the union tag is `group_already_deleted`.

Return type bool

class dropbox.team.**GroupFullInfo**(*group_name=None*, *group_id=None*, *group_management_type=None*, *created=None*, *group_external_id=None*, *member_count=None*, *members=None*)
Bases: `dropbox.team_common.GroupSummary`

Full description of a group.

Variables

- ***members*** – List of group members.
- ***created*** – The group creation time as a UTC timestamp in milliseconds since the Unix epoch.

__init__(*group_name=None*, *group_id=None*, *group_management_type=None*, *created=None*, *group_external_id=None*, *member_count=None*, *members=None*)

created
The group creation time as a UTC timestamp in milliseconds since the Unix epoch.

Return type long

members
List of group members.

Return type list of [GroupMemberInfo]

class dropbox.team.**GroupMemberInfo**(*profile=None*, *access_type=None*)
Bases: `object`

Profile of group member, and role in group.

Variables

- ***profile*** – Profile of group member.
- ***access_type*** – The role that the user has in the group.

__init__(*profile=None*, *access_type=None*)

access_type
The role that the user has in the group.

Return type *GroupAccessType*

profile
Profile of group member.

Return type *MemberProfile*

class dropbox.team.**GroupMemberSelector**(*group=None*, *user=None*)
Bases: `object`

Argument for selecting a group and a single user.

Variables

- ***group*** – Specify a group.
- ***user*** – Identity of a user that is a member of `group`.

__init__(*group=None*, *user=None*)

group

Specify a group.

Return type *GroupSelector*

user

Identity of a user that is a member of `group`.

Return type *UserSelectorArg*

class dropbox.team.**GroupMemberSelectorError**(*tag*, *value=None*)

Bases: *dropbox.team.GroupSelectorError*

Error that can be raised when *GroupMemberSelector* is used, and the user is required to be a member of the specified group.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***member_not_in_group*** – The specified user is not a member of this group.

is_member_not_in_group()

Check if the union tag is `member_not_in_group`.

Return type bool

member_not_in_group = GroupMemberSelectorError('member_not_in_group', None)

class dropbox.team.**GroupMemberSetAccessTypeError**(*tag*, *value=None*)

Bases: *dropbox.team.GroupMemberSelectorError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***user_cannot_be_manager_of_company_managed_group*** – A company managed group cannot be managed by a user.

is_user_cannot_be_manager_of_company_managed_group()

Check if the union tag is `user_cannot_be_manager_of_company_managed_group`.

Return type bool

user_cannot_be_manager_of_company_managed_group = GroupMemberSetAccessTypeError('user_cannot_be

class dropbox.team.**GroupMembersAddArg**(*group=None*, *members=None*, *return_members=None*)

Bases: *dropbox.team.IncludeMembersArg*

Variables

- ***group*** – Group to which users will be added.
- ***members*** – List of users to be added to the group.

__init__(*group=None*, *members=None*, *return_members=None*)

group

Group to which users will be added.

Return type *GroupSelector*

members
List of users to be added to the group.

Return type list of [MemberAccess]

class dropbox.team.**GroupMembersAddError**(*tag*, *value=None*)
Bases: *dropbox.team.GroupSelectorError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***duplicate_user*** – You cannot add duplicate users. One or more of the members you are trying to add is already a member of the group.
- ***group_not_in_team*** – Group is not in this team. You cannot add members to a group that is outside of your team.
- **of [str] members_not_in_team** (*list*) – These members are not part of your team. Currently, you cannot add members to a group if they are not part of your team, though this may change in a subsequent version. To add new members to your Dropbox Business team, use the **:route:'members/add'** endpoint.
- **of [str] users_not_found** (*list*) – These users were not found in Dropbox.
- ***user_must_be_active_to_be_owner*** – A suspended user cannot be added to a group as `GroupAccessType.owner`.
- **of [str] user_cannot_be_manager_of_company_managed_group** (*list*) – A company-managed group cannot be managed by a user.

duplicate_user = GroupMembersAddError('duplicate_user', None)

get_members_not_in_team()
These members are not part of your team. Currently, you cannot add members to a group if they are not part of your team, though this may change in a subsequent version. To add new members to your Dropbox Business team, use the `dropbox.dropbox.Dropbox.team_members_add()` endpoint.

Only call this if *is_members_not_in_team()* is true.

Return type list of [str]

get_user_cannot_be_manager_of_company_managed_group()
A company-managed group cannot be managed by a user.

Only call this if *is_user_cannot_be_manager_of_company_managed_group()* is true.

Return type list of [str]

get_users_not_found()
These users were not found in Dropbox.

Only call this if *is_users_not_found()* is true.

Return type list of [str]

group_not_in_team = GroupMembersAddError('group_not_in_team', None)

is_duplicate_user()
Check if the union tag is `duplicate_user`.

Return type bool

is_group_not_in_team()
Check if the union tag is `group_not_in_team`.

Return type bool

is_members_not_in_team()
Check if the union tag is `members_not_in_team`.

Return type bool

is_user_cannot_be_manager_of_company_managed_group()
Check if the union tag is `user_cannot_be_manager_of_company_managed_group`.

Return type bool

is_user_must_be_active_to_be_owner()
Check if the union tag is `user_must_be_active_to_be_owner`.

Return type bool

is_users_not_found()
Check if the union tag is `users_not_found`.

Return type bool

classmethod members_not_in_team(*val*)
Create an instance of this class set to the `members_not_in_team` tag with value `val`.

Parameters of [str] val (*list*) –

Return type *GroupMembersAddError*

classmethod user_cannot_be_manager_of_company_managed_group(*val*)
Create an instance of this class set to the `user_cannot_be_manager_of_company_managed_group` tag with value `val`.

Parameters of [str] val (*list*) –

Return type *GroupMembersAddError*

user_must_be_active_to_be_owner = GroupMembersAddError('user_must_be_active_to_be_owner', None)

classmethod users_not_found(*val*)
Create an instance of this class set to the `users_not_found` tag with value `val`.

Parameters of [str] val (*list*) –

Return type *GroupMembersAddError*

class dropbox.team.**GroupMembersChangeResult**(*group_info=None*, *async_job_id=None*)
Bases: `object`

Result returned by `dropbox.dropbox.Dropbox.team_groups_members_add()` and `dropbox.dropbox.Dropbox.team_groups_members_remove()`.

Variables

- ***group_info*** – The group info after member change operation has been performed.
- ***async_job_id*** – An ID that can be used to obtain the status of granting/revoking group-owned resources.

__init__(*group_info=None*, *async_job_id=None*)

async_job_id
An ID that can be used to obtain the status of granting/revoking group-owned resources.

Return type str

group_info
The group info after member change operation has been performed.

Return type *GroupFullInfo*

class dropbox.team.**GroupMembersRemoveArg**(*group=None*, *users=None*, *return_members=None*)
Bases: *dropbox.team.IncludeMembersArg*

Variables

- ***group*** – Group from which users will be removed.
- ***users*** – List of users to be removed from the group.

__init__(*group=None*, *users=None*, *return_members=None*)

group
Group from which users will be removed.

Return type *GroupSelector*

users
List of users to be removed from the group.

Return type list of [UserSelectorArg]

class dropbox.team.**GroupMembersRemoveError**(*tag*, *value=None*)
Bases: *dropbox.team.GroupMembersSelectorError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***group_not_in_team*** – Group is not in this team. You cannot remove members from a group that is outside of your team.

group_not_in_team = GroupMembersRemoveError('group_not_in_team', None)

is_group_not_in_team()
Check if the union tag is `group_not_in_team`.

Return type bool

class dropbox.team.**GroupMembersSelector**(*group=None*, *users=None*)
Bases: `object`

Argument for selecting a group and a list of users.

Variables

- ***group*** – Specify a group.
- ***users*** – A list of users that are members of `group`.

__init__(*group=None*, *users=None*)

group
Specify a group.

Return type *GroupSelector*

users
A list of users that are members of `group`.

Return type *UsersSelectorArg*

class dropbox.team.**GroupMembersSelectorError**(*tag*, *value=None*)
Bases: *dropbox.team.GroupSelectorError*

Error that can be raised when *GroupMembersSelector* is used, and the users are required to be members of the specified group.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***member_not_in_group*** – At least one of the specified users is not a member of the group.

is_member_not_in_group()
Check if the union tag is `member_not_in_group`.

Return type bool

member_not_in_group = GroupMembersSelectorError('member_not_in_group', None)

class dropbox.team.**GroupMembersSetAccessTypeArg**(*group=None*, *user=None*, *access_type=None*, *return_members=None*)
Bases: *dropbox.team.GroupMemberSelector*

Variables

- ***access_type*** – New group access type the user will have.
- ***return_members*** – Whether to return the list of members in the group. Note that the default value will cause all the group members to be returned in the response. This may take a long time for large groups.

__init__(*group=None*, *user=None*, *access_type=None*, *return_members=None*)

access_type
New group access type the user will have.

Return type *GroupAccessType*

return_members
Whether to return the list of members in the group. Note that the default value will cause all the group members to be returned in the response. This may take a long time for large groups.

Return type bool

class dropbox.team.**GroupSelector**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Argument for selecting a single group, either by group_id or by external group ID.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***group_id*** (*str*) – Group ID.
- ***group_external_id*** (*str*) – External ID of the group.

get_group_external_id()
External ID of the group.

Only call this if *is_group_external_id()* is true.

Return type str

get_group_id()
Group ID.

Only call this if `is_group_id()` is true.

Return type str

classmethod group_external_id(*val*)
Create an instance of this class set to the `group_external_id` tag with value `val`.

Parameters **val** (*str*) –

Return type *GroupSelector*

classmethod group_id(*val*)
Create an instance of this class set to the `group_id` tag with value `val`.

Parameters **val** (*str*) –

Return type *GroupSelector*

is_group_external_id()
Check if the union tag is `group_external_id`.

Return type bool

is_group_id()
Check if the union tag is `group_id`.

Return type bool

class dropbox.team.**GroupSelectorError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Error that can be raised when `GroupSelector` is used.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***group_not_found*** – No matching group found. No groups match the specified group ID.

group_not_found = GroupSelectorError('group_not_found', None)

is_group_not_found()
Check if the union tag is `group_not_found`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

other = GroupSelectorError('other', None)

class dropbox.team.**GroupUpdateArgs**(*group=None*, *return_members=None*, *new_group_name=None*, *new_group_external_id=None*, *new_group_management_type=None*)
Bases: `dropbox.team.IncludeMembersArg`

Variables

- ***group*** – Specify a group.
- ***new_group_name*** – Optional argument. Set group name to this if provided.

- ***new_group_external_id*** – Optional argument. New group external ID. If the argument is None, the group's external_id won't be updated. If the argument is empty string, the group's external id will be cleared.
- ***new_group_management_type*** – Set new group management type, if provided.

__init__(*group=None*, *return_members=None*, *new_group_name=None*, *new_group_external_id=None*, *new_group_management_type=None*)

group
Specify a group.

Return type *GroupSelector*

new_group_external_id
Optional argument. New group external ID. If the argument is None, the group's external_id won't be updated. If the argument is empty string, the group's external id will be cleared.

Return type str

new_group_management_type
Set new group management type, if provided.

Return type team_common.GroupManagementType_validator

new_group_name
Optional argument. Set group name to this if provided.

Return type str

class dropbox.team.**GroupUpdateError**(*tag*, *value=None*)
Bases: *dropbox.team.GroupSelectorError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***external_id_already_in_use*** – The new external ID is already being used by another group.

external_id_already_in_use = GroupUpdateError('external_id_already_in_use', None)

is_external_id_already_in_use()
Check if the union tag is `external_id_already_in_use`.

Return type bool

class dropbox.team.**GroupsGetInfoError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***group_not_on_team*** – The group is not on your team.

group_not_on_team = GroupsGetInfoError('group_not_on_team', None)

is_group_not_on_team()
Check if the union tag is `group_not_on_team`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

other = GroupsGetInfoError('other', None)

class dropbox.team.**GroupsGetInfoItem**(*tag*, *value=None*)

Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables

- ***id_not_found*** (*str*) – An ID that was provided as a parameter to **:route:'groups/get_info'**, and did not match a corresponding group. The ID can be a group ID, or an external ID, depending on how the method was called.
- ***group_info*** (GroupFullInfo) – Info about a group.

get_group_info()

Info about a group.

Only call this if *is_group_info()* is true.

Return type *GroupFullInfo*

get_id_not_found()

An ID that was provided as a parameter to dropbox.dropbox.Dropbox.team_groups_get_info(), and did not match a corresponding group. The ID can be a group ID, or an external ID, depending on how the method was called.

Only call this if *is_id_not_found()* is true.

Return type str

classmethod group_info(*val*)

Create an instance of this class set to the group_info tag with value val.

Parameters **val** (GroupFullInfo) –

Return type *GroupsGetInfoItem*

classmethod id_not_found(*val*)

Create an instance of this class set to the id_not_found tag with value val.

Parameters **val** (*str*) –

Return type *GroupsGetInfoItem*

is_group_info()

Check if the union tag is group_info.

Return type bool

is_id_not_found()

Check if the union tag is id_not_found.

Return type bool

class dropbox.team.**GroupsListArg**(*limit=None*)

Bases: object

Variables ***limit*** – Number of results to return per call.

__init__(*limit=None*)

limit

Number of results to return per call.

Return type long

class `dropbox.team.`**`GroupsListContinueArg`**(*cursor=None*)
Bases: `object`

Variables ***cursor*** – Indicates from what point to get the next set of groups.

`__init__`(*cursor=None*)

`cursor`
Indicates from what point to get the next set of groups.

Return type str

class `dropbox.team.`**`GroupsListContinueError`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***invalid_cursor*** – The cursor is invalid.

`invalid_cursor` = GroupsListContinueError('invalid_cursor', None)

`is_invalid_cursor`()
Check if the union tag is `invalid_cursor`.

Return type bool

`is_other`()
Check if the union tag is `other`.

Return type bool

`other` = GroupsListContinueError('other', None)

class `dropbox.team.`**`GroupsListResult`**(*groups=None*, *cursor=None*, *has_more=None*)
Bases: `object`

Variables

- ***cursor*** – Pass the cursor into `dropbox.dropbox.Dropbox.team_groups_list_continue()` to obtain the additional groups.
- ***has_more*** – Is true if there are additional groups that have not been returned yet. An additional call to `dropbox.dropbox.Dropbox.team_groups_list_continue()` can retrieve them.

`__init__`(*groups=None*, *cursor=None*, *has_more=None*)

`cursor`
Pass the cursor into `dropbox.dropbox.Dropbox.team_groups_list_continue()` to obtain the additional groups.

Return type str

`groups`

Return type list of [team_common.GroupSummary_validator]

`has_more`
Is true if there are additional groups that have not been returned yet. An additional call to `dropbox.dropbox.Dropbox.team_groups_list_continue()` can retrieve them.

Return type bool

class `dropbox.team.`**`GroupsMembersListArg`**(*group=None*, *limit=None*)
Bases: `object`

Variables

- ***group*** – The group whose members are to be listed.
- ***limit*** – Number of results to return per call.

__init__(*group=None*, *limit=None*)

group
The group whose members are to be listed.

Return type *GroupSelector*

limit
Number of results to return per call.

Return type long

class dropbox.team.**GroupsMembersListContinueArg**(*cursor=None*)
Bases: object

Variables ***cursor*** – Indicates from what point to get the next set of groups.

__init__(*cursor=None*)

cursor
Indicates from what point to get the next set of groups.

Return type str

class dropbox.team.**GroupsMembersListContinueError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***invalid_cursor*** – The cursor is invalid.

invalid_cursor = GroupsMembersListContinueError('invalid_cursor', None)

is_invalid_cursor()
Check if the union tag is `invalid_cursor`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

other = GroupsMembersListContinueError('other', None)

class dropbox.team.**GroupsMembersListResult**(*members=None*, *cursor=None*, *has_more=None*)
Bases: object

Variables

- ***cursor*** – Pass the cursor into `dropbox.dropbox.Dropbox.team_groups_members_list_continu` to obtain additional group members.
- ***has_more*** – Is true if there are additional group members that have not been returned yet. An additional call to `dropbox.dropbox.Dropbox.team_groups_members_list_continue()` can retrieve them.

__init__(*members=None*, *cursor=None*, *has_more=None*)

cursor
Pass the cursor into `dropbox.dropbox.Dropbox.team_groups_members_list_continue()` to obtain additional group members.

Return type str

has_more
Is true if there are additional group members that have not been returned yet. An additional call to `dropbox.dropbox.Dropbox.team_groups_members_list_continue()` can retrieve them.

Return type bool

members

Return type list of [GroupMemberInfo]

class `dropbox.team.`**GroupsPollError**(*tag*, *value=None*)
Bases: *dropbox.async.PollError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***access_denied*** – You are not allowed to poll this job.

access_denied = GroupsPollError('access_denied', None)

is_access_denied()
Check if the union tag is `access_denied`.

Return type bool

class `dropbox.team.`**GroupsSelector**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Argument for selecting a list of groups, either by group_ids, or external group IDs.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- **of [str] group_ids** (*list*) – List of group IDs.
- **of [str] group_external_ids** (*list*) – List of external IDs of groups.

get_group_external_ids()
List of external IDs of groups.

Only call this if *is_group_external_ids()* is true.

Return type list of [str]

get_group_ids()
List of group IDs.

Only call this if *is_group_ids()* is true.

Return type list of [str]

classmethod group_external_ids(*val*)
Create an instance of this class set to the `group_external_ids` tag with value `val`.

Parameters **of [str] val** (*list*) –

Return type *GroupsSelector*

classmethod group_ids(*val*)
Create an instance of this class set to the `group_ids` tag with value `val`.

Parameters of [str] val(*list*) –

Return type *GroupsSelector*

is_group_external_ids()
Check if the union tag is `group_external_ids`.

Return type bool

is_group_ids()
Check if the union tag is `group_ids`.

Return type bool

class dropbox.team.**IncludeMembersArg**(*return_members=None*)
Bases: `object`

Variables ***return_members*** – Whether to return the list of members in the group. Note that the default value will cause all the group members to be returned in the response. This may take a long time for large groups.

__init__(*return_members=None*)

return_members
Whether to return the list of members in the group. Note that the default value will cause all the group members to be returned in the response. This may take a long time for large groups.

Return type bool

class dropbox.team.**ListMemberAppsArg**(*team_member_id=None*)
Bases: `object`

Variables ***team_member_id*** – The team member id

__init__(*team_member_id=None*)

team_member_id
The team member id

Return type str

class dropbox.team.**ListMemberAppsError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Error returned by `dropbox.dropbox.Dropbox.team_linked_apps_list_member_linked_apps()`.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***member_not_found*** – Member not found.

is_member_not_found()
Check if the union tag is `member_not_found`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

member_not_found = ListMemberAppsError('member_not_found', None)

other = ListMemberAppsError('other', None)

class `dropbox.team.`**`ListMemberAppsResult`**(*linked_api_apps=None*)

Bases: `object`

Variables ***linked_api_apps*** – List of third party applications linked by this team member

`__init__`(*linked_api_apps=None*)

`linked_api_apps`

List of third party applications linked by this team member

Return type list of [ApiApp]

class `dropbox.team.`**`ListMemberDevicesArg`**(*team_member_id=None*, *include_web_sessions=None*, *include_desktop_clients=None*, *include_mobile_clients=None*)

Bases: `object`

Variables

- ***team_member_id*** – The team's member id
- ***include_web_sessions*** – Whether to list web sessions of the team's member
- ***include_desktop_clients*** – Whether to list linked desktop devices of the team's member
- ***include_mobile_clients*** – Whether to list linked mobile devices of the team's member

`__init__`(*team_member_id=None*, *include_web_sessions=None*, *include_desktop_clients=None*, *include_mobile_clients=None*)

`include_desktop_clients`

Whether to list linked desktop devices of the team's member

Return type bool

`include_mobile_clients`

Whether to list linked mobile devices of the team's member

Return type bool

`include_web_sessions`

Whether to list web sessions of the team's member

Return type bool

`team_member_id`

The team's member id

Return type str

class `dropbox.team.`**`ListMemberDevicesError`**(*tag*, *value=None*)

Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***member_not_found*** – Member not found.

`is_member_not_found`()

Check if the union tag is `member_not_found`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

member_not_found = ListMemberDevicesError('member_not_found', None)

other = ListMemberDevicesError('other', None)

class dropbox.team.**ListMemberDevicesResult**(*active_web_sessions=None, desktop_client_sessions=None, mobile_client_sessions=None*)

Bases: `object`

Variables

- ***active_web_sessions*** – List of web sessions made by this team member
- ***desktop_client_sessions*** – List of desktop clients used by this team member
- ***mobile_client_sessions*** – List of mobile client used by this team member

__init__(*active_web_sessions=None, desktop_client_sessions=None, mobile_client_sessions=None*)

active_web_sessions
List of web sessions made by this team member

Return type list of [ActiveWebSession]

desktop_client_sessions
List of desktop clients used by this team member

Return type list of [DesktopClientSession]

mobile_client_sessions
List of mobile client used by this team member

Return type list of [MobileClientSession]

class dropbox.team.**ListMembersAppsArg**(*cursor=None*)

Bases: `object`

Arguments for `dropbox.dropbox.Dropbox.team_linked_apps_list_members_linked_apps()`.

Variables ***cursor*** – At the first call to the `dropbox.dropbox.Dropbox.team_linked_apps_list_members_` the cursor shouldn't be passed. Then, if the result of the call includes a cursor, the following requests should include the received cursors in order to receive the next sub list of the team applications

__init__(*cursor=None*)

cursor
At the first call to the `dropbox.dropbox.Dropbox.team_linked_apps_list_members_linked_apps()` the cursor shouldn't be passed. Then, if the result of the call includes a cursor, the following requests should include the received cursors in order to receive the next sub list of the team applications

Return type str

class dropbox.team.**ListMembersAppsError**(*tag, value=None*)

Bases: `dropbox.stone_base.Union`

Error returned by `dropbox.dropbox.Dropbox.team_linked_apps_list_members_linked_apps()`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***reset*** – Indicates that the cursor has been invalidated. Call `dropbox.dropbox.Dropbox.team_linked_apps_list_members_linked_apps()` again with an empty cursor to obtain a new cursor.

is_other()
Check if the union tag is `other`.

Return type bool

is_reset()
Check if the union tag is `reset`.

Return type bool

other = ListMembersAppsError('other', None)

reset = ListMembersAppsError('reset', None)

class `dropbox.team.`**ListMembersAppsResult**(*apps=None*, *has_more=None*, *cursor=None*)
Bases: `object`

Information returned by `dropbox.dropbox.Dropbox.team_linked_apps_list_members_linked_apps()`.

Variables

- ***apps*** – The linked applications of each member of the team
- ***has_more*** – If true, then there are more apps available. Pass the cursor to `dropbox.dropbox.Dropbox.team_linked_apps_list_members_linked_apps()` to retrieve the rest.
- ***cursor*** – Pass the cursor into `dropbox.dropbox.Dropbox.team_linked_apps_list_members_li` to receive the next sub list of team's applications.

__init__(*apps=None*, *has_more=None*, *cursor=None*)

apps
The linked applications of each member of the team

Return type list of [MemberLinkedApps]

cursor
Pass the cursor into `dropbox.dropbox.Dropbox.team_linked_apps_list_members_linked_apps()` to receive the next sub list of team's applications.

Return type str

has_more
If true, then there are more apps available. Pass the cursor to `dropbox.dropbox.Dropbox.team_linked_apps_list_members_linked_apps()` to retrieve the rest.

Return type bool

class `dropbox.team.`**ListMembersDevicesArg**(*cursor=None*, *include_web_sessions=None*, *include_desktop_clients=None*, *include_mobile_clients=None*)
Bases: `object`

Variables

- ***cursor*** – At the first call to the `dropbox.dropbox.Dropbox.team_devices_list_members_devic` the cursor shouldn't be passed. Then, if the result of the call includes a cursor, the following requests should include the received cursors in order to receive the next sub list of team devices

- ***include_web_sessions*** – Whether to list web sessions of the team members
- ***include_desktop_clients*** – Whether to list desktop clients of the team members
- ***include_mobile_clients*** – Whether to list mobile clients of the team members

__init__(*cursor=None*, *include_web_sessions=None*, *include_desktop_clients=None*, *include_mobile_clients=None*)

cursor

At the first call to the `dropbox.dropbox.Dropbox.team_devices_list_members_devices()` the cursor shouldn't be passed. Then, if the result of the call includes a cursor, the following requests should include the received cursors in order to receive the next sub list of team devices

Return type str

include_desktop_clients

Whether to list desktop clients of the team members

Return type bool

include_mobile_clients

Whether to list mobile clients of the team members

Return type bool

include_web_sessions

Whether to list web sessions of the team members

Return type bool

class `dropbox.team.`**ListMembersDevicesError**(*tag*, *value=None*)

Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***reset*** – Indicates that the cursor has been invalidated. Call `dropbox.dropbox.Dropbox.team_devices_list_members_devices()` again with an empty cursor to obtain a new cursor.

is_other()

Check if the union tag is `other`.

Return type bool

is_reset()

Check if the union tag is `reset`.

Return type bool

other = ListMembersDevicesError('other', None)

reset = ListMembersDevicesError('reset', None)

class `dropbox.team.`**ListMembersDevicesResult**(*devices=None*, *has_more=None*, *cursor=None*)

Bases: `object`

Variables

- ***devices*** – The devices of each member of the team
- ***has_more*** – If true, then there are more devices available. Pass the cursor to `dropbox.dropbox.Dropbox.team_devices_list_members_devices()` to retrieve the rest.

- ***cursor*** – Pass the cursor into `dropbox.dropbox.Dropbox.team_devices_list_members_device` to receive the next sub list of team's devices.

__init__(*devices=None*, *has_more=None*, *cursor=None*)

cursor
Pass the cursor into `dropbox.dropbox.Dropbox.team_devices_list_members_devices()` to receive the next sub list of team's devices.

Return type str

devices
The devices of each member of the team

Return type list of [MemberDevices]

has_more
If true, then there are more devices available. Pass the cursor to `dropbox.dropbox.Dropbox.team_devices_list_members_devices()` to retrieve the rest.

Return type bool

class `dropbox.team.`**ListTeamAppsArg**(*cursor=None*)
Bases: `object`

Arguments for `dropbox.dropbox.Dropbox.team_linked_apps_list_team_linked_apps()`.

Variables ***cursor*** – At the first call to the `dropbox.dropbox.Dropbox.team_linked_apps_list_team_lin` the cursor shouldn't be passed. Then, if the result of the call includes a cursor, the following requests should include the received cursors in order to receive the next sub list of the team applications

__init__(*cursor=None*)

cursor
At the first call to the `dropbox.dropbox.Dropbox.team_linked_apps_list_team_linked_apps()` the cursor shouldn't be passed. Then, if the result of the call includes a cursor, the following requests should include the received cursors in order to receive the next sub list of the team applications

Return type str

class `dropbox.team.`**ListTeamAppsError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Error returned by `dropbox.dropbox.Dropbox.team_linked_apps_list_team_linked_apps()`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***reset*** – Indicates that the cursor has been invalidated. Call `dropbox.dropbox.Dropbox.team_linked_apps_list_team_linked_apps()` again with an empty cursor to obtain a new cursor.

is_other()
Check if the union tag is `other`.

Return type bool

is_reset()
Check if the union tag is `reset`.

Return type bool

other = ListTeamAppsError('other', None)

reset = ListTeamAppsError('reset', None)

class dropbox.team.**ListTeamAppsResult**(*apps=None*, *has_more=None*, *cursor=None*)

Bases: object

Information returned by dropbox.dropbox.Dropbox.team_linked_apps_list_team_linked_apps().

Variables

- ***apps*** – The linked applications of each member of the team
- ***has_more*** – If true, then there are more apps available. Pass the cursor to dropbox.dropbox.Dropbox.team_linked_apps_list_team_linked_apps() to retrieve the rest.
- ***cursor*** – Pass the cursor into dropbox.dropbox.Dropbox.team_linked_apps_list_team_linke to receive the next sub list of team's applications.

__init__(*apps=None*, *has_more=None*, *cursor=None*)

apps

The linked applications of each member of the team

Return type list of [MemberLinkedApps]

cursor

Pass the cursor into dropbox.dropbox.Dropbox.team_linked_apps_list_team_linked_apps() to receive the next sub list of team's applications.

Return type str

has_more

If true, then there are more apps available. Pass the cursor to dropbox.dropbox.Dropbox.team_linked_apps_list_team_linked_apps() to retrieve the rest.

Return type bool

class dropbox.team.**ListTeamDevicesArg**(*cursor=None*, *include_web_sessions=None*, *include_desktop_clients=None*, *include_mobile_clients=None*)

Bases: object

Variables

- ***cursor*** – At the first call to the dropbox.dropbox.Dropbox.team_devices_list_team_devices(the cursor shouldn't be passed. Then, if the result of the call includes a cursor, the following requests should include the received cursors in order to receive the next sub list of team devices
- ***include_web_sessions*** – Whether to list web sessions of the team members
- ***include_desktop_clients*** – Whether to list desktop clients of the team members
- ***include_mobile_clients*** – Whether to list mobile clients of the team members

__init__(*cursor=None*, *include_web_sessions=None*, *include_desktop_clients=None*, *include_mobile_clients=None*)

cursor

At the first call to the dropbox.dropbox.Dropbox.team_devices_list_team_devices() the cursor shouldn't be passed. Then, if the result of the call includes a cursor, the following requests should include the received cursors in order to receive the next sub list of team devices

Return type str

include_desktop_clients
Whether to list desktop clients of the team members

Return type bool

include_mobile_clients
Whether to list mobile clients of the team members

Return type bool

include_web_sessions
Whether to list web sessions of the team members

Return type bool

class `dropbox.team.`**`ListTeamDevicesError`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***reset*** – Indicates that the cursor has been invalidated. Call `dropbox.dropbox.Dropbox.team_devices_list_team_devices()` again with an empty cursor to obtain a new cursor.

is_other()
Check if the union tag is `other`.

Return type bool

is_reset()
Check if the union tag is `reset`.

Return type bool

other = ListTeamDevicesError('other', None)

reset = ListTeamDevicesError('reset', None)

class `dropbox.team.`**`ListTeamDevicesResult`**(*devices=None*, *has_more=None*, *cursor=None*)
Bases: `object`

Variables

- ***devices*** – The devices of each member of the team
- ***has_more*** – If true, then there are more devices available. Pass the cursor to `dropbox.dropbox.Dropbox.team_devices_list_team_devices()` to retrieve the rest.
- ***cursor*** – Pass the cursor into `dropbox.dropbox.Dropbox.team_devices_list_team_devices()` to receive the next sub list of team's devices.

__init__(*devices=None*, *has_more=None*, *cursor=None*)

cursor
Pass the cursor into `dropbox.dropbox.Dropbox.team_devices_list_team_devices()` to receive the next sub list of team's devices.

Return type str

devices
The devices of each member of the team

Return type list of [MemberDevices]

has_more
If true, then there are more devices available. Pass the cursor to `dropbox.dropbox.Dropbox.team_devices_list_team_devices()` to retrieve the rest.

Return type bool

class `dropbox.team.`**MemberAccess**(*user=None*, *access_type=None*)
Bases: `object`

Specify access type a member should have when joined to a group.

Variables

- ***user*** – Identity of a user.
- ***access_type*** – Access type.

__init__(*user=None*, *access_type=None*)

access_type
Access type.

Return type *GroupAccessType*

user
Identity of a user.

Return type *UserSelectorArg*

class `dropbox.team.`**MemberAddArg**(*member_email=None*, *member_given_name=None*, *member_surname=None*, *member_external_id=None*, *send_welcome_email=None*, *role=None*)
Bases: `object`

Variables

- ***member_given_name*** – Member's first name.
- ***member_surname*** – Member's last name.
- ***member_external_id*** – External ID for member.
- ***send_welcome_email*** – Whether to send a welcome email to the member. If send_welcome_email is false, no email invitation will be sent to the user. This may be useful for apps using single sign-on (SSO) flows for onboarding that want to handle announcements themselves.

__init__(*member_email=None*, *member_given_name=None*, *member_surname=None*, *member_external_id=None*, *send_welcome_email=None*, *role=None*)

member_email

Return type str

member_external_id
External ID for member.

Return type str

member_given_name
Member's first name.

Return type str

member_surname
Member's last name.

Return type str

role

Return type *AdminTier*

send_welcome_email

Whether to send a welcome email to the member. If send_welcome_email is false, no email invitation will be sent to the user. This may be useful for apps using single sign-on (SSO) flows for onboarding that want to handle announcements themselves.

Return type bool

class `dropbox.team.`**MemberAddResult**(*tag*, *value=None*)

Bases: `dropbox.stone_base.Union`

Describes the result of attempting to add a single user to the team. 'success' is the only value indicating that a user was indeed added to the team - the other values explain the type of failure that occurred, and include the email of the user for which the operation has failed.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***success*** (`TeamMemberInfo`) – Describes a user that was successfully added to the team.
- ***team_license_limit*** (*str*) – Team is already full. The organization has no available licenses.
- ***free_team_member_limit_reached*** (*str*) – Team is already full. The free team member limit has been reached.
- ***user_already_on_team*** (*str*) – User is already on this team. The provided email address is associated with a user who is already a member of (including in recoverable state) or invited to the team.
- ***user_on_another_team*** (*str*) – User is already on another team. The provided email address is associated with a user that is already a member or invited to another team.
- ***user_already_paired*** (*str*) – User is already paired.
- ***user_migration_failed*** (*str*) – User migration has failed.
- ***duplicate_external_member_id*** (*str*) – A user with the given external member ID already exists on the team (including in recoverable state).
- ***user_creation_failed*** (*str*) – User creation has failed.

classmethod duplicate_external_member_id(*val*)

Create an instance of this class set to the `duplicate_external_member_id` tag with value `val`.

Parameters **val** (*str*) –

Return type *MemberAddResult*

classmethod free_team_member_limit_reached(*val*)

Create an instance of this class set to the `free_team_member_limit_reached` tag with value `val`.

Parameters **val** (*str*) –

Return type *MemberAddResult*

get_duplicate_external_member_id()
A user with the given external member ID already exists on the team (including in recoverable state).

Only call this if *is_duplicate_external_member_id()* is true.

Return type str

get_free_team_member_limit_reached()
Team is already full. The free team member limit has been reached.

Only call this if *is_free_team_member_limit_reached()* is true.

Return type str

get_success()
Describes a user that was successfully added to the team.

Only call this if *is_success()* is true.

Return type *TeamMemberInfo*

get_team_license_limit()
Team is already full. The organization has no available licenses.

Only call this if *is_team_license_limit()* is true.

Return type str

get_user_already_on_team()
User is already on this team. The provided email address is associated with a user who is already a member of (including in recoverable state) or invited to the team.

Only call this if *is_user_already_on_team()* is true.

Return type str

get_user_already_paired()
User is already paired.

Only call this if *is_user_already_paired()* is true.

Return type str

get_user_creation_failed()
User creation has failed.

Only call this if *is_user_creation_failed()* is true.

Return type str

get_user_migration_failed()
User migration has failed.

Only call this if *is_user_migration_failed()* is true.

Return type str

get_user_on_another_team()
User is already on another team. The provided email address is associated with a user that is already a member or invited to another team.

Only call this if *is_user_on_another_team()* is true.

Return type str

is_duplicate_external_member_id()
Check if the union tag is `duplicate_external_member_id`.

Return type bool

is_free_team_member_limit_reached()
Check if the union tag is `free_team_member_limit_reached`.

Return type bool

is_success()
Check if the union tag is `success`.

Return type bool

is_team_license_limit()
Check if the union tag is `team_license_limit`.

Return type bool

is_user_already_on_team()
Check if the union tag is `user_already_on_team`.

Return type bool

is_user_already_paired()
Check if the union tag is `user_already_paired`.

Return type bool

is_user_creation_failed()
Check if the union tag is `user_creation_failed`.

Return type bool

is_user_migration_failed()
Check if the union tag is `user_migration_failed`.

Return type bool

is_user_on_another_team()
Check if the union tag is `user_on_another_team`.

Return type bool

classmethod success(*val*)
Create an instance of this class set to the `success` tag with value `val`.

Parameters **val** (`TeamMemberInfo`) –

Return type *MemberAddResult*

classmethod team_license_limit(*val*)
Create an instance of this class set to the `team_license_limit` tag with value `val`.

Parameters **val** (*str*) –

Return type *MemberAddResult*

classmethod user_already_on_team(*val*)
Create an instance of this class set to the `user_already_on_team` tag with value `val`.

Parameters **val** (*str*) –

Return type *MemberAddResult*

classmethod user_already_paired(*val*)
Create an instance of this class set to the `user_already_paired` tag with value `val`.

Parameters **val** (*str*) –

Return type *MemberAddResult*

classmethod user_creation_failed(*val*)
Create an instance of this class set to the `user_creation_failed` tag with value `val`.

Parameters **val** (*str*) –

Return type *MemberAddResult*

classmethod user_migration_failed(*val*)
Create an instance of this class set to the `user_migration_failed` tag with value `val`.

Parameters **val** (*str*) –

Return type *MemberAddResult*

classmethod user_on_another_team(*val*)
Create an instance of this class set to the `user_on_another_team` tag with value `val`.

Parameters **val** (*str*) –

Return type *MemberAddResult*

class dropbox.team.**MemberDevices**(*team_member_id=None*, *web_sessions=None*, *desktop_clients=None*, *mobile_clients=None*)
Bases: `object`

Information on devices of a team's member.

Variables

- ***team_member_id*** – The member unique Id
- ***web_sessions*** – List of web sessions made by this team member
- ***desktop_clients*** – List of desktop clients by this team member
- ***mobile_clients*** – List of mobile clients by this team member

__init__(*team_member_id=None*, *web_sessions=None*, *desktop_clients=None*, *mobile_clients=None*)

desktop_clients
List of desktop clients by this team member

Return type list of [DesktopClientSession]

mobile_clients
List of mobile clients by this team member

Return type list of [MobileClientSession]

team_member_id
The member unique Id

Return type str

web_sessions
List of web sessions made by this team member

Return type list of [ActiveWebSession]

class dropbox.team.**MemberLinkedApps**(*team_member_id=None*, *linked_api_apps=None*)
Bases: `object`

Information on linked applications of a team member.

Variables

- ***team_member_id*** – The member unique Id
- ***linked_api_apps*** – List of third party applications linked by this team member

__init__(*team_member_id=None*, *linked_api_apps=None*)

linked_api_apps

List of third party applications linked by this team member

Return type list of [ApiApp]

team_member_id

The member unique Id

Return type str

class dropbox.team.**MemberProfile**(*team_member_id=None*, *email=None*, *email_verified=None*, *status=None*, *name=None*, *membership_type=None*, *external_id=None*, *account_id=None*)

Bases: object

Basic member profile.

Variables

- ***team_member_id*** – ID of user as a member of a team.
- ***external_id*** – External ID that a team can attach to the user. An application using the API may find it easier to use their own IDs instead of Dropbox IDs like account_id or team_member_id.
- ***account_id*** – A user's account identifier.
- ***email*** – Email address of user.
- ***email_verified*** – Is true if the user's email is verified to be owned by the user.
- ***status*** – The user's status as a member of a specific team.
- ***name*** – Representations for a person's name.
- ***membership_type*** – The user's membership type: full (normal team member) vs limited (does not use a license; no access to the team's shared quota).

__init__(*team_member_id=None*, *email=None*, *email_verified=None*, *status=None*, *name=None*, *membership_type=None*, *external_id=None*, *account_id=None*)

account_id

A user's account identifier.

Return type str

email

Email address of user.

Return type str

email_verified

Is true if the user's email is verified to be owned by the user.

Return type bool

external_id

External ID that a team can attach to the user. An application using the API may find it easier to use their own IDs instead of Dropbox IDs like account_id or team_member_id.

Return type str

membership_type
The user's membership type: full (normal team member) vs limited (does not use a license; no access to the team's shared quota).

Return type *TeamMembershipType*

name
Representations for a person's name.

Return type users.Name_validator

status
The user's status as a member of a specific team.

Return type *TeamMemberStatus*

team_member_id
ID of user as a member of a team.

Return type str

class dropbox.team.**MemberSelectorError**(*tag*, *value=None*)
Bases: *dropbox.team.UserSelectorError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***user_not_in_team*** – The user is not a member of the team.

is_user_not_in_team()
Check if the union tag is `user_not_in_team`.

Return type bool

user_not_in_team = MemberSelectorError('user_not_in_team', None)

class dropbox.team.**MembersAddArg**(*new_members=None*, *force_async=None*)
Bases: `object`

Variables

- ***new_members*** – Details of new members to be added to the team.
- ***force_async*** – Whether to force the add to happen asynchronously.

__init__(*new_members=None*, *force_async=None*)

force_async
Whether to force the add to happen asynchronously.

Return type bool

new_members
Details of new members to be added to the team.

Return type list of [MemberAddArg]

class dropbox.team.**MembersAddJobStatus**(*tag*, *value=None*)
Bases: *dropbox.async.PollResultBase*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- **of [MemberAddResult] complete** (*list*) – The asynchronous job has finished. For each member that was specified in the parameter **:type:'MembersAddArg'** that was provided to **:route:'members/add'**, a corresponding item is returned in this list.
- ***failed*** (*str*) – The asynchronous job returned an error. The string contains an error message.

classmethod complete(*val*)

Create an instance of this class set to the `complete` tag with value `val`.

Parameters **of [MemberAddResult] val** (*list*) –

Return type *MembersAddJobStatus*

classmethod failed(*val*)

Create an instance of this class set to the `failed` tag with value `val`.

Parameters **val** (*str*) –

Return type *MembersAddJobStatus*

get_complete()

The asynchronous job has finished. For each member that was specified in the parameter *MembersAddArg* that was provided to `dropbox.dropbox.Dropbox.team_members_add()`, a corresponding item is returned in this list.

Only call this if *is_complete()* is true.

Return type list of [MemberAddResult]

get_failed()

The asynchronous job returned an error. The string contains an error message.

Only call this if *is_failed()* is true.

Return type str

is_complete()

Check if the union tag is `complete`.

Return type bool

is_failed()

Check if the union tag is `failed`.

Return type bool

class `dropbox.team.`**MembersAddLaunch**(*tag*, *value=None*)

Bases: *dropbox.async.LaunchResultBase*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

classmethod complete(*val*)

Create an instance of this class set to the `complete` tag with value `val`.

Parameters **of [MemberAddResult] val** (*list*) –

Return type *MembersAddLaunch*

get_complete()

Only call this if *is_complete()* is true.

Return type list of [MemberAddResult]

is_complete()
Check if the union tag is `complete`.

Return type bool

class dropbox.team.**MembersDeactivateArg**(*user=None*, *wipe_data=None*)
Bases: `object`

Exactly one of team_member_id, email, or external_id must be provided to identify the user account.

Variables

- ***user*** – Identity of user to remove/suspend.
- ***wipe_data*** – If provided, controls if the user's data will be deleted on their linked devices.

__init__(*user=None*, *wipe_data=None*)

user
Identity of user to remove/suspend.

Return type *UserSelectorArg*

wipe_data
If provided, controls if the user's data will be deleted on their linked devices.

Return type bool

class dropbox.team.**MembersDeactivateError**(*tag*, *value=None*)
Bases: *dropbox.team.UserSelectorError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***user_not_in_team*** – The user is not a member of the team.

is_other()
Check if the union tag is `other`.

Return type bool

is_user_not_in_team()
Check if the union tag is `user_not_in_team`.

Return type bool

other = MembersDeactivateError('other', None)

user_not_in_team = MembersDeactivateError('user_not_in_team', None)

class dropbox.team.**MembersGetInfoArgs**(*members=None*)
Bases: `object`

Variables ***members*** – List of team members.

__init__(*members=None*)

members
List of team members.

Return type list of [UserSelectorArg]

class dropbox.team.**MembersGetInfoError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

is_other()
Check if the union tag is `other`.

Return type bool

other = MembersGetInfoError('other', None)

class dropbox.team.**MembersGetInfoItem**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

Describes a result obtained for a single user whose id was specified in the parameter of dropbox.dropbox.Dropbox.team_members_get_info().

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***id_not_found*** (*str*) – An ID that was provided as a parameter to **:route:'members/get_info'**, and did not match a corresponding user. This might be a team_member_id, an email, or an external ID, depending on how the method was called.
- ***member_info*** (TeamMemberInfo) – Info about a team member.

get_id_not_found()
An ID that was provided as a parameter to dropbox.dropbox.Dropbox.team_members_get_info(), and did not match a corresponding user. This might be a team_member_id, an email, or an external ID, depending on how the method was called.

Only call this if *is_id_not_found()* is true.

Return type str

get_member_info()
Info about a team member.

Only call this if *is_member_info()* is true.

Return type *TeamMemberInfo*

classmethod id_not_found(*val*)
Create an instance of this class set to the `id_not_found` tag with value `val`.

Parameters **val** (*str*) –

Return type *MembersGetInfoItem*

is_id_not_found()
Check if the union tag is `id_not_found`.

Return type bool

is_member_info()
Check if the union tag is `member_info`.

Return type bool

classmethod member_info(*val*)
Create an instance of this class set to the `member_info` tag with value `val`.

Parameters **val** (TeamMemberInfo) –

Return type *MembersGetInfoItem*

class dropbox.team.**MembersListArg**(*limit=None*, *include_removed=None*)
Bases: object

Variables

- ***limit*** – Number of results to return per call.
- ***include_removed*** – Whether to return removed members.

__init__(*limit=None*, *include_removed=None*)

include_removed
Whether to return removed members.

Return type bool

limit
Number of results to return per call.

Return type long

class dropbox.team.**MembersListContinueArg**(*cursor=None*)
Bases: object

Variables ***cursor*** – Indicates from what point to get the next set of members.

__init__(*cursor=None*)

cursor
Indicates from what point to get the next set of members.

Return type str

class dropbox.team.**MembersListContinueError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

Variables ***invalid_cursor*** – The cursor is invalid.

invalid_cursor = MembersListContinueError('invalid_cursor', None)

is_invalid_cursor()
Check if the union tag is invalid_cursor.

Return type bool

is_other()
Check if the union tag is other.

Return type bool

other = MembersListContinueError('other', None)

class dropbox.team.**MembersListError**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the is_* methods will return true. To get the associated value of a tag (if one exists), use the corresponding get_* method.

is_other()
Check if the union tag is other.

Return type bool

other = MembersListError('other', None)

class dropbox.team.**MembersListResult**(*members=None*, *cursor=None*, *has_more=None*)
Bases: object

Variables

- ***members*** – List of team members.
- ***cursor*** – Pass the cursor into `dropbox.dropbox.Dropbox.team_members_list_continue()` to obtain the additional members.
- ***has_more*** – Is true if there are additional team members that have not been returned yet. An additional call to `dropbox.dropbox.Dropbox.team_members_list_continue()` can retrieve them.

__init__(*members=None*, *cursor=None*, *has_more=None*)

cursor

Pass the cursor into `dropbox.dropbox.Dropbox.team_members_list_continue()` to obtain the additional members.

Return type str

has_more

Is true if there are additional team members that have not been returned yet. An additional call to `dropbox.dropbox.Dropbox.team_members_list_continue()` can retrieve them.

Return type bool

members

List of team members.

Return type list of [TeamMemberInfo]

class `dropbox.team.`**MembersRecoverArg**(*user=None*)

Bases: `object`

Exactly one of team_member_id, email, or external_id must be provided to identify the user account.

Variables ***user*** – Identity of user to recover.

__init__(*user=None*)

user

Identity of user to recover.

Return type *UserSelectorArg*

class `dropbox.team.`**MembersRecoverError**(*tag*, *value=None*)

Bases: *`dropbox.team.UserSelectorError`*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***user_unrecoverable*** – The user is not recoverable.
- ***user_not_in_team*** – The user is not a member of the team.
- ***team_license_limit*** – Team is full. The organization has no available licenses.

is_other()

Check if the union tag is `other`.

Return type bool

is_team_license_limit()

Check if the union tag is `team_license_limit`.

Return type bool

is_user_not_in_team()
Check if the union tag is `user_not_in_team`.

Return type bool

is_user_unrecoverable()
Check if the union tag is `user_unrecoverable`.

Return type bool

other = MembersRecoverError('other', None)

team_license_limit = MembersRecoverError('team_license_limit', None)

user_not_in_team = MembersRecoverError('user_not_in_team', None)

user_unrecoverable = MembersRecoverError('user_unrecoverable', None)

class dropbox.team.**MembersRemoveArg**(*user=None*, *wipe_data=None*, *transfer_dest_id=None*, *transfer_admin_id=None*, *keep_account=None*)
Bases: *dropbox.team.MembersDeactivateArg*

Variables

- ***transfer_dest_id*** – If provided, files from the deleted member account will be transferred to this user.
- ***transfer_admin_id*** – If provided, errors during the transfer process will be sent via email to this user. If the transfer_dest_id argument was provided, then this argument must be provided as well.
- ***keep_account*** – Downgrade the member to a Basic account. The user will retain the email address associated with their Dropbox account and data in their account that is not restricted to team members.

__init__(*user=None*, *wipe_data=None*, *transfer_dest_id=None*, *transfer_admin_id=None*, *keep_account=None*)

keep_account
Downgrade the member to a Basic account. The user will retain the email address associated with their Dropbox account and data in their account that is not restricted to team members.

Return type bool

transfer_admin_id
If provided, errors during the transfer process will be sent via email to this user. If the transfer_dest_id argument was provided, then this argument must be provided as well.

Return type *UserSelectorArg*

transfer_dest_id
If provided, files from the deleted member account will be transferred to this user.

Return type *UserSelectorArg*

class dropbox.team.**MembersRemoveError**(*tag*, *value=None*)
Bases: *dropbox.team.MembersDeactivateError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***remove_last_admin*** – The user is the last admin of the team, so it cannot be removed from it.
- ***removed_and_transfer_dest_should_differ*** – Expected removed user and transfer_dest user to be different
- ***removed_and_transfer_admin_should_differ*** – Expected removed user and transfer_admin user to be different.
- ***transfer_dest_user_not_found*** – No matching user found for the argument transfer_dest_id.
- ***transfer_dest_user_not_in_team*** – The provided transfer_dest_id does not exist on this team.
- ***transfer_admin_user_not_found*** – No matching user found for the argument transfer_admin_id.
- ***transfer_admin_user_not_in_team*** – The provided transfer_admin_id does not exist on this team.
- ***unspecified_transfer_admin_id*** – The transfer_admin_id argument must be provided when file transfer is requested.
- ***transfer_admin_is_not_admin*** – Specified transfer_admin user is not a team admin.
- ***cannot_keep_account_and_transfer*** – Cannot keep account and transfer the data to another user at the same time.
- ***cannot_keep_account_and_delete_data*** – Cannot keep account and delete the data at the same time.
- ***email_address_too_long_to_be_disabled*** – The email address of the user is too long to be disabled.

cannot_keep_account_and_delete_data = MembersRemoveError('cannot_keep_account_and_delete_data', Nor

cannot_keep_account_and_transfer = MembersRemoveError('cannot_keep_account_and_transfer', None)

email_address_too_long_to_be_disabled = MembersRemoveError('email_address_too_long_to_be_disabled',

is_cannot_keep_account_and_delete_data()

Check if the union tag is `cannot_keep_account_and_delete_data`.

Return type bool

is_cannot_keep_account_and_transfer()

Check if the union tag is `cannot_keep_account_and_transfer`.

Return type bool

is_email_address_too_long_to_be_disabled()

Check if the union tag is `email_address_too_long_to_be_disabled`.

Return type bool

is_remove_last_admin()

Check if the union tag is `remove_last_admin`.

Return type bool

is_removed_and_transfer_admin_should_differ()

Check if the union tag is `removed_and_transfer_admin_should_differ`.

Return type bool

is_removed_and_transfer_dest_should_differ()
Check if the union tag is `removed_and_transfer_dest_should_differ`.

Return type bool

is_transfer_admin_is_not_admin()
Check if the union tag is `transfer_admin_is_not_admin`.

Return type bool

is_transfer_admin_user_not_found()
Check if the union tag is `transfer_admin_user_not_found`.

Return type bool

is_transfer_admin_user_not_in_team()
Check if the union tag is `transfer_admin_user_not_in_team`.

Return type bool

is_transfer_dest_user_not_found()
Check if the union tag is `transfer_dest_user_not_found`.

Return type bool

is_transfer_dest_user_not_in_team()
Check if the union tag is `transfer_dest_user_not_in_team`.

Return type bool

is_unspecified_transfer_admin_id()
Check if the union tag is `unspecified_transfer_admin_id`.

Return type bool

remove_last_admin = MembersRemoveError('remove_last_admin', None)

removed_and_transfer_admin_should_differ = MembersRemoveError('removed_and_transfer_admin_should

removed_and_transfer_dest_should_differ = MembersRemoveError('removed_and_transfer_dest_should_di

transfer_admin_is_not_admin = MembersRemoveError('transfer_admin_is_not_admin', None)

transfer_admin_user_not_found = MembersRemoveError('transfer_admin_user_not_found', None)

transfer_admin_user_not_in_team = MembersRemoveError('transfer_admin_user_not_in_team', None)

transfer_dest_user_not_found = MembersRemoveError('transfer_dest_user_not_found', None)

transfer_dest_user_not_in_team = MembersRemoveError('transfer_dest_user_not_in_team', None)

unspecified_transfer_admin_id = MembersRemoveError('unspecified_transfer_admin_id', None)

class dropbox.team.**MembersSendWelcomeError**(*tag*, *value=None*)
Bases: *dropbox.team.MemberSelectorError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

is_other()
Check if the union tag is `other`.

Return type bool

other = MembersSendWelcomeError('other', None)

class `dropbox.team.`**`MembersSetPermissionsArg`**(*user=None*, *new_role=None*)
Bases: `object`

Exactly one of team_member_id, email, or external_id must be provided to identify the user account.

Variables

- ***user*** – Identity of user whose role will be set.
- ***new_role*** – The new role of the member.

`__init__`(*user=None*, *new_role=None*)

`new_role`
The new role of the member.

Return type *AdminTier*

`user`
Identity of user whose role will be set.

Return type *UserSelectorArg*

class `dropbox.team.`**`MembersSetPermissionsError`**(*tag*, *value=None*)
Bases: *`dropbox.team.UserSelectorError`*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***last_admin*** – Cannot remove the admin setting of the last admin.
- ***user_not_in_team*** – The user is not a member of the team.
- ***cannot_set_permissions*** – Cannot remove/grant permissions.
- ***team_license_limit*** – Team is full. The organization has no available licenses.

`cannot_set_permissions` = MembersSetPermissionsError('cannot_set_permissions', None)

`is_cannot_set_permissions`()
Check if the union tag is `cannot_set_permissions`.

Return type bool

`is_last_admin`()
Check if the union tag is `last_admin`.

Return type bool

`is_other`()
Check if the union tag is `other`.

Return type bool

`is_team_license_limit`()
Check if the union tag is `team_license_limit`.

Return type bool

`is_user_not_in_team`()
Check if the union tag is `user_not_in_team`.

Return type bool

`last_admin` = MembersSetPermissionsError('last_admin', None)

other = MembersSetPermissionsError('other', None)

team_license_limit = MembersSetPermissionsError('team_license_limit', None)

user_not_in_team = MembersSetPermissionsError('user_not_in_team', None)

class dropbox.team.**MembersSetPermissionsResult**(*team_member_id=None*, *role=None*)

Bases: object

Variables

- ***team_member_id*** – The member ID of the user to which the change was applied.
- ***role*** – The role after the change.

__init__(*team_member_id=None*, *role=None*)

role

The role after the change.

Return type *AdminTier*

team_member_id

The member ID of the user to which the change was applied.

Return type str

class dropbox.team.**MembersSetProfileArg**(*user=None*, *new_email=None*, *new_external_id=None*, *new_given_name=None*, *new_surname=None*)

Bases: object

Exactly one of team_member_id, email, or external_id must be provided to identify the user account. At least one of new_email, new_external_id, new_given_name, and/or new_surname must be provided.

Variables

- ***user*** – Identity of user whose profile will be set.
- ***new_email*** – New email for member.
- ***new_external_id*** – New external ID for member.
- ***new_given_name*** – New given name for member.
- ***new_surname*** – New surname for member.

__init__(*user=None*, *new_email=None*, *new_external_id=None*, *new_given_name=None*, *new_surname=None*)

new_email

New email for member.

Return type str

new_external_id

New external ID for member.

Return type str

new_given_name

New given name for member.

Return type str

new_surname

New surname for member.

Return type str

user
Identity of user whose profile will be set.

Return type *UserSelectorArg*

class dropbox.team.**MembersSetProfileError**(*tag*, *value=None*)
Bases: *dropbox.team.MemberSelectorError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***external_id_and_new_external_id_unsafe*** – It is unsafe to use both external_id and new_external_id
- ***no_new_data_specified*** – None of new_email, new_given_name, new_surname, or new_external_id are specified
- ***email_reserved_for_other_user*** – Email is already reserved for another user.
- ***external_id_used_by_other_user*** – The external ID is already in use by another team member.
- ***set_profile_disallowed*** – Setting profile disallowed
- ***param_cannot_be_empty*** – Parameter new_email cannot be empty.

email_reserved_for_other_user = MembersSetProfileError('email_reserved_for_other_user', None)

external_id_and_new_external_id_unsafe = MembersSetProfileError('external_id_and_new_external_id_uns

external_id_used_by_other_user = MembersSetProfileError('external_id_used_by_other_user', None)

is_email_reserved_for_other_user()
Check if the union tag is `email_reserved_for_other_user`.

Return type bool

is_external_id_and_new_external_id_unsafe()
Check if the union tag is `external_id_and_new_external_id_unsafe`.

Return type bool

is_external_id_used_by_other_user()
Check if the union tag is `external_id_used_by_other_user`.

Return type bool

is_no_new_data_specified()
Check if the union tag is `no_new_data_specified`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_param_cannot_be_empty()
Check if the union tag is `param_cannot_be_empty`.

Return type bool

is_set_profile_disallowed()
Check if the union tag is `set_profile_disallowed`.

Return type bool

no_new_data_specified = MembersSetProfileError('no_new_data_specified', None)

other = MembersSetProfileError('other', None)

param_cannot_be_empty = MembersSetProfileError('param_cannot_be_empty', None)

set_profile_disallowed = MembersSetProfileError('set_profile_disallowed', None)

class dropbox.team.**MembersSuspendError**(*tag*, *value=None*)
Bases: *dropbox.team.MembersDeactivateError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***suspend_inactive_user*** – The user is not active, so it cannot be suspended.
- ***suspend_last_admin*** – The user is the last admin of the team, so it cannot be suspended.
- ***team_license_limit*** – Team is full. The organization has no available licenses.

is_suspend_inactive_user()
Check if the union tag is `suspend_inactive_user`.

Return type bool

is_suspend_last_admin()
Check if the union tag is `suspend_last_admin`.

Return type bool

is_team_license_limit()
Check if the union tag is `team_license_limit`.

Return type bool

suspend_inactive_user = MembersSuspendError('suspend_inactive_user', None)

suspend_last_admin = MembersSuspendError('suspend_last_admin', None)

team_license_limit = MembersSuspendError('team_license_limit', None)

class dropbox.team.**MembersUnsuspendArg**(*user=None*)
Bases: `object`

Exactly one of team_member_id, email, or external_id must be provided to identify the user account.

Variables ***user*** – Identity of user to unsuspend.

__init__(*user=None*)

user
Identity of user to unsuspend.

Return type *UserSelectorArg*

class dropbox.team.**MembersUnsuspendError**(*tag*, *value=None*)
Bases: *dropbox.team.MembersDeactivateError*

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***unsuspend_non_suspended_member*** – The user is unsuspended, so it cannot be unsuspended again.
- ***team_license_limit*** – Team is full. The organization has no available licenses.

is_team_license_limit()
Check if the union tag is `team_license_limit`.

Return type bool

is_unsuspend_non_suspended_member()
Check if the union tag is `unsuspend_non_suspended_member`.

Return type bool

team_license_limit = MembersUnsuspendError('team_license_limit', None)

unsuspend_non_suspended_member = MembersUnsuspendError('unsuspend_non_suspended_member', None)

class dropbox.team.**MobileClientPlatform**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***iphone*** – Official Dropbox iPhone client
- ***ipad*** – Official Dropbox iPad client
- ***android*** – Official Dropbox Android client
- ***windows_phone*** – Official Dropbox Windows phone client
- ***blackberry*** – Official Dropbox Blackberry client

android = MobileClientPlatform('android', None)

blackberry = MobileClientPlatform('blackberry', None)

ipad = MobileClientPlatform('ipad', None)

iphone = MobileClientPlatform('iphone', None)

is_android()
Check if the union tag is `android`.

Return type bool

is_blackberry()
Check if the union tag is `blackberry`.

Return type bool

is_ipad()
Check if the union tag is `ipad`.

Return type bool

is_iphone()
Check if the union tag is `iphone`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_windows_phone()
Check if the union tag is `windows_phone`.

Return type bool

other = MobileClientPlatform('other', None)

windows_phone = MobileClientPlatform('windows_phone', None)

class dropbox.team.**MobileClientSession**(*session_id=None*, *device_name=None*, *client_type=None*, *ip_address=None*, *country=None*, *created=None*, *updated=None*, *client_version=None*, *os_version=None*, *last_carrier=None*)
Bases: *dropbox.team.DeviceSession*

Information about linked Dropbox mobile client sessions

Variables

- ***device_name*** – The device name
- ***client_type*** – The mobile application type
- ***client_version*** – The dropbox client version
- ***os_version*** – The hosting OS version
- ***last_carrier*** – last carrier used by the device

__init__(*session_id=None*, *device_name=None*, *client_type=None*, *ip_address=None*, *country=None*, *created=None*, *updated=None*, *client_version=None*, *os_version=None*, *last_carrier=None*)

client_type
The mobile application type

Return type *MobileClientPlatform*

client_version
The dropbox client version

Return type str

device_name
The device name

Return type str

last_carrier
last carrier used by the device

Return type str

os_version
The hosting OS version

Return type str

class dropbox.team.**RemovedStatus**(*is_recoverable=None*)
Bases: `object`

Variables ***is_recoverable*** – True if the removed team member is recoverable

__init__(*is_recoverable=None*)

is_recoverable
True if the removed team member is recoverable

Return type bool

class dropbox.team.**RevokeDesktopClientArg**(*session_id=None*, *team_member_id=None*, *delete_on_unlink=None*)
Bases: *dropbox.team.DeviceSessionArg*

Variables ***delete_on_unlink*** – Whether to delete all files of the account (this is possible only if supported by the desktop client and will be made the next time the client access the account)

__init__(*session_id=None*, *team_member_id=None*, *delete_on_unlink=None*)

delete_on_unlink
Whether to delete all files of the account (this is possible only if supported by the desktop client and will be made the next time the client access the account)

Return type bool

class dropbox.team.**RevokeDeviceSessionArg**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***web_session*** (DeviceSessionArg) – End an active session
- ***desktop_client*** (RevokeDesktopClientArg) – Unlink a linked desktop device
- ***mobile_client*** (DeviceSessionArg) – Unlink a linked mobile device

classmethod desktop_client(*val*)
Create an instance of this class set to the `desktop_client` tag with value `val`.

Parameters **val** (RevokeDesktopClientArg) –

Return type *RevokeDeviceSessionArg*

get_desktop_client()
Unlink a linked desktop device

Only call this if *is_desktop_client()* is true.

Return type *RevokeDesktopClientArg*

get_mobile_client()
Unlink a linked mobile device

Only call this if *is_mobile_client()* is true.

Return type *DeviceSessionArg*

get_web_session()
End an active session

Only call this if *is_web_session()* is true.

Return type *DeviceSessionArg*

is_desktop_client()
Check if the union tag is `desktop_client`.

Return type bool

is_mobile_client()
Check if the union tag is `mobile_client`.

Return type bool

is_web_session()
Check if the union tag is `web_session`.

Return type bool

classmethod mobile_client(*val*)
Create an instance of this class set to the `mobile_client` tag with value `val`.

Parameters **val** (`DeviceSessionArg`) –

Return type *RevokeDeviceSessionArg*

classmethod web_session(*val*)
Create an instance of this class set to the `web_session` tag with value `val`.

Parameters **val** (`DeviceSessionArg`) –

Return type *RevokeDeviceSessionArg*

class dropbox.team.**RevokeDeviceSessionBatchArg**(*revoke_devices=None*)
Bases: `object`

__init__(*revoke_devices=None*)

revoke_devices

Return type list of [RevokeDeviceSessionArg]

class dropbox.team.**RevokeDeviceSessionBatchError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

is_other()
Check if the union tag is `other`.

Return type bool

other = RevokeDeviceSessionBatchError('other', None)

class dropbox.team.**RevokeDeviceSessionBatchResult**(*revoke_devices_status=None*)
Bases: `object`

__init__(*revoke_devices_status=None*)

revoke_devices_status

Return type list of [RevokeDeviceSessionStatus]

class dropbox.team.**RevokeDeviceSessionError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***device_session_not_found*** – Device session not found.
- ***member_not_found*** – Member not found.

device_session_not_found = RevokeDeviceSessionError('device_session_not_found', None)

is_device_session_not_found()
Check if the union tag is `device_session_not_found`.

Return type bool

is_member_not_found()
Check if the union tag is `member_not_found`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

member_not_found = RevokeDeviceSessionError('member_not_found', None)

other = RevokeDeviceSessionError('other', None)

class dropbox.team.**RevokeDeviceSessionStatus**(*success=None*, *error_type=None*)
Bases: `object`

Variables

- ***success*** – Result of the revoking request
- ***error_type*** – The error cause in case of a failure

__init__(*success=None*, *error_type=None*)

error_type
The error cause in case of a failure

Return type *RevokeDeviceSessionError*

success
Result of the revoking request

Return type bool

class dropbox.team.**RevokeLinkedApiAppArg**(*app_id=None*, *team_member_id=None*, *keep_app_folder=None*)
Bases: `object`

Variables

- ***app_id*** – The application's unique id
- ***team_member_id*** – The unique id of the member owning the device
- ***keep_app_folder*** – Whether to keep the application dedicated folder (in case the application uses one)

__init__(*app_id=None*, *team_member_id=None*, *keep_app_folder=None*)

app_id
The application's unique id

Return type str

keep_app_folder
Whether to keep the application dedicated folder (in case the application uses one)

Return type bool

team_member_id
The unique id of the member owning the device

Return type str

class `dropbox.team.`**`RevokeLinkedApiAppBatchArg`**(*revoke_linked_app=None*)
Bases: `object`

`__init__`(*revoke_linked_app=None*)

`revoke_linked_app`

Return type list of [RevokeLinkedApiAppArg]

class `dropbox.team.`**`RevokeLinkedAppBatchError`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Error returned by `dropbox.dropbox.Dropbox.team_linked_apps_revoke_linked_app_batch()`.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

`is_other`()
Check if the union tag is `other`.

Return type bool

`other` = RevokeLinkedAppBatchError('other', None)

class `dropbox.team.`**`RevokeLinkedAppBatchResult`**(*revoke_linked_app_status=None*)
Bases: `object`

`__init__`(*revoke_linked_app_status=None*)

`revoke_linked_app_status`

Return type list of [RevokeLinkedAppStatus]

class `dropbox.team.`**`RevokeLinkedAppError`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Error returned by `dropbox.dropbox.Dropbox.team_linked_apps_revoke_linked_app()`.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***`app_not_found`*** – Application not found.
- ***`member_not_found`*** – Member not found.

`app_not_found` = RevokeLinkedAppError('app_not_found', None)

`is_app_not_found`()
Check if the union tag is `app_not_found`.

Return type bool

`is_member_not_found`()
Check if the union tag is `member_not_found`.

Return type bool

`is_other`()
Check if the union tag is `other`.

Return type bool

`member_not_found` = RevokeLinkedAppError('member_not_found', None)

other = RevokeLinkedAppError('other', None)

class dropbox.team.**RevokeLinkedAppStatus**(*success=None*, *error_type=None*)

Bases: object

Variables

- ***success*** – Result of the revoking request
- ***error_type*** – The error cause in case of a failure

__init__(*success=None*, *error_type=None*)

error_type

The error cause in case of a failure

Return type *RevokeLinkedAppError*

success

Result of the revoking request

Return type bool

class dropbox.team.**StorageBucket**(*bucket=None*, *users=None*)

Bases: object

Describes the number of users in a specific storage bucket.

Variables

- ***bucket*** – The name of the storage bucket. For example, '1G' is a bucket of users with storage size up to 1 Giga.
- ***users*** – The number of people whose storage is in the range of this storage bucket.

__init__(*bucket=None*, *users=None*)

bucket

The name of the storage bucket. For example, '1G' is a bucket of users with storage size up to 1 Giga.

Return type str

users

The number of people whose storage is in the range of this storage bucket.

Return type long

class dropbox.team.**TeamGetInfoResult**(*name=None*, *team_id=None*, *num_licensed_users=None*, *num_provisioned_users=None*, *policies=None*)

Bases: object

Variables

- ***name*** – The name of the team.
- ***team_id*** – The ID of the team.
- ***num_licensed_users*** – The number of licenses available to the team.
- ***num_provisioned_users*** – The number of accounts that have been invited or are already active members of the team.

__init__(*name=None*, *team_id=None*, *num_licensed_users=None*, *num_provisioned_users=None*, *policies=None*)

name

The name of the team.

Return type str

num_licensed_users
The number of licenses available to the team.

Return type long

num_provisioned_users
The number of accounts that have been invited or are already active members of the team.

Return type long

policies

Return type team_policies.TeamMemberPolicies_validator

team_id
The ID of the team.

Return type str

class dropbox.team.**TeamMemberInfo**(*profile=None*, *role=None*)
Bases: object

Information about a team member.

Variables

- ***profile*** – Profile of a user as a member of a team.
- ***role*** – The user's role in the team.

__init__(*profile=None*, *role=None*)

profile
Profile of a user as a member of a team.

Return type *TeamMemberProfile*

role
The user's role in the team.

Return type *AdminTier*

class dropbox.team.**TeamMemberProfile**(*team_member_id=None*, *email=None*, *email_verified=None*, *status=None*, *name=None*, *membership_type=None*, *groups=None*, *external_id=None*, *account_id=None*)
Bases: dropbox.team.MemberProfile

Profile of a user as a member of a team.

Variables ***groups*** – List of group IDs of groups that the user belongs to.

__init__(*team_member_id=None*, *email=None*, *email_verified=None*, *status=None*, *name=None*, *membership_type=None*, *groups=None*, *external_id=None*, *account_id=None*)

groups
List of group IDs of groups that the user belongs to.

Return type list of [str]

class dropbox.team.**TeamMemberStatus**(*tag*, *value=None*)
Bases: dropbox.stone_base.Union

The user's status as a member of a specific team.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***active*** – User has successfully joined the team.
- ***invited*** – User has been invited to a team, but has not joined the team yet.
- ***suspended*** – User is no longer a member of the team, but the account can be unsuspended, re-establishing the user as a team member.
- ***removed*** (`RemovedStatus`) – User is no longer a member of the team. Removed users are only listed when include_removed is true in members/list.

active = TeamMemberStatus('active', None)

get_removed()

User is no longer a member of the team. Removed users are only listed when include_removed is true in members/list.

Only call this if *is_removed()* is true.

Return type *RemovedStatus*

invited = TeamMemberStatus('invited', None)

is_active()

Check if the union tag is `active`.

Return type bool

is_invited()

Check if the union tag is `invited`.

Return type bool

is_removed()

Check if the union tag is `removed`.

Return type bool

is_suspended()

Check if the union tag is `suspended`.

Return type bool

classmethod removed(*val*)

Create an instance of this class set to the `removed` tag with value `val`.

Parameters **val** (`RemovedStatus`) –

Return type *TeamMemberStatus*

suspended = TeamMemberStatus('suspended', None)

class `dropbox.team.`**TeamMembershipType**(*tag*, *value=None*)

Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***full*** – User uses a license and has full access to team resources like the shared quota.

- ***limited*** – User does not have access to the shared quota and team admins have restricted administrative control.

full = TeamMembershipType('full', None)

is_full()
Check if the union tag is `full`.

Return type bool

is_limited()
Check if the union tag is `limited`.

Return type bool

limited = TeamMembershipType('limited', None)

class dropbox.team.**UpdatePropertyTemplateArg**(*template_id=None*, *name=None*, *description=None*, *add_fields=None*)

Bases: `object`

Variables

- ***template_id*** – An identifier for property template added by `dropbox.dropbox.Dropbox.team_properties_template_add()`.
- ***name*** – A display name for the property template. Property template names can be up to 256 bytes.
- ***description*** – Description for new property template. Property template descriptions can be up to 1024 bytes.
- ***add_fields*** – This is a list of custom properties to add to the property template. There can be up to 64 properties in a single property template.

__init__(*template_id=None*, *name=None*, *description=None*, *add_fields=None*)

add_fields
This is a list of custom properties to add to the property template. There can be up to 64 properties in a single property template.

Return type list of [properties.PropertyFieldTemplate_validator]

description
Description for new property template. Property template descriptions can be up to 1024 bytes.

Return type str

name
A display name for the property template. Property template names can be up to 256 bytes.

Return type str

template_id
An identifier for property template added by `dropbox.dropbox.Dropbox.team_properties_template_add(`

Return type str

class dropbox.team.**UpdatePropertyTemplateResult**(*template_id=None*)

Bases: `object`

Variables ***template_id*** – An identifier for property template added by `dropbox.dropbox.Dropbox.team_properties_template_add()`.

__init__(*template_id=None*)

template_id
An identifier for property template added by `dropbox.dropbox.Dropbox.team_properties_template_add()`

Return type str

class `dropbox.team.`**`UserSelectorArg`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Argument for selecting a single user, either by team_member_id, external_id or email.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

classmethod email(*val*)
Create an instance of this class set to the `email` tag with value `val`.

Parameters **val** (*str*) –

Return type *UserSelectorArg*

classmethod external_id(*val*)
Create an instance of this class set to the `external_id` tag with value `val`.

Parameters **val** (*str*) –

Return type *UserSelectorArg*

get_email()
Only call this if *is_email()* is true.

Return type str

get_external_id()
Only call this if *is_external_id()* is true.

Return type str

get_team_member_id()
Only call this if *is_team_member_id()* is true.

Return type str

is_email()
Check if the union tag is `email`.

Return type bool

is_external_id()
Check if the union tag is `external_id`.

Return type bool

is_team_member_id()
Check if the union tag is `team_member_id`.

Return type bool

classmethod team_member_id(*val*)
Create an instance of this class set to the `team_member_id` tag with value `val`.

Parameters **val** (*str*) –

Return type *UserSelectorArg*

class `dropbox.team.`**`UserSelectorError`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Error that can be returned whenever a struct derived from *UserSelectorArg* is used.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***user_not_found*** – No matching user found. The provided team_member_id, email, or external_id does not exist on this team.

is_user_not_found()
Check if the union tag is `user_not_found`.

Return type bool

user_not_found = UserSelectorError('user_not_found', None)

class dropbox.team.**UsersSelectorArg**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

Argument for selecting a list of users, either by team_member_ids, external_ids or emails.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- **of [str] team_member_ids** (*list*) – List of member IDs.
- **of [str] external_ids** (*list*) – List of external user IDs.
- **of [str] emails** (*list*) – List of email addresses.

classmethod emails(*val*)
Create an instance of this class set to the `emails` tag with value `val`.

Parameters **of [str] val** (*list*) –

Return type *UsersSelectorArg*

classmethod external_ids(*val*)
Create an instance of this class set to the `external_ids` tag with value `val`.

Parameters **of [str] val** (*list*) –

Return type *UsersSelectorArg*

get_emails()
List of email addresses.

Only call this if *is_emails()* is true.

Return type list of [str]

get_external_ids()
List of external user IDs.

Only call this if *is_external_ids()* is true.

Return type list of [str]

get_team_member_ids()
List of member IDs.

Only call this if *is_team_member_ids()* is true.

Return type list of [str]

is_emails()
Check if the union tag is `emails`.

Return type bool

is_external_ids()
Check if the union tag is `external_ids`.

Return type bool

is_team_member_ids()
Check if the union tag is `team_member_ids`.

Return type bool

classmethod team_member_ids(*val*)
Create an instance of this class set to the `team_member_ids` tag with value `val`.

Parameters of [str] val(*list*) –

Return type *UsersSelectorArg*

2.7 dropbox.users – Users

This namespace contains endpoints and data types for user management.

class dropbox.users.**Account**(*account_id=None*, *name=None*, *email=None*, *email_verified=None*, *disabled=None*, *profile_photo_url=None*)
Bases: `object`

The amount of detail revealed about an account depends on the user being queried and the user making the query.

Variables

- ***account_id*** – The user's unique Dropbox ID.
- ***name*** – Details of a user's name.
- ***email*** – The user's e-mail address. Do not rely on this without checking the `email_verified` field. Even then, it's possible that the user has since lost access to their e-mail.
- ***email_verified*** – Whether the user has verified their e-mail address.
- ***profile_photo_url*** – URL for the photo representing the user, if one is set.
- ***disabled*** – Whether the user has been disabled.

__init__(*account_id=None*, *name=None*, *email=None*, *email_verified=None*, *disabled=None*, *profile_photo_url=None*)

account_id
The user's unique Dropbox ID.

Return type str

disabled
Whether the user has been disabled.

Return type bool

email
The user's e-mail address. Do not rely on this without checking the `email_verified` field. Even then, it's possible that the user has since lost access to their e-mail.

Return type str

email_verified
Whether the user has verified their e-mail address.

Return type bool

name
Details of a user's name.

Return type *Name*

profile_photo_url
URL for the photo representing the user, if one is set.

Return type str

class `dropbox.users.`**`AccountType`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

What type of account this user has.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***basic*** – The basic account type.
- ***pro*** – The Dropbox Pro account type.
- ***business*** – The Dropbox Business account type.

basic = AccountType('basic', None)

business = AccountType('business', None)

is_basic()
Check if the union tag is `basic`.

Return type bool

is_business()
Check if the union tag is `business`.

Return type bool

is_pro()
Check if the union tag is `pro`.

Return type bool

pro = AccountType('pro', None)

class `dropbox.users.`**`BasicAccount`**(*account_id=None*, *name=None*, *email=None*, *email_verified=None*, *disabled=None*, *is_teammate=None*, *profile_photo_url=None*, *team_member_id=None*)
Bases: *`dropbox.users.Account`*

Basic information about any account.

Variables

- ***is_teammate*** – Whether this user is a teammate of the current user. If this account is the current user's account, then this will be `True`.
- ***team_member_id*** – The user's unique team member id. This field will only be present if the user is part of a team and `is_teammate` is `True`.

__init__(*account_id=None*, *name=None*, *email=None*, *email_verified=None*, *disabled=None*, *is_teammate=None*, *profile_photo_url=None*, *team_member_id=None*)

is_teammate

Whether this user is a teammate of the current user. If this account is the current user's account, then this will be `True`.

Return type bool

team_member_id

The user's unique team member id. This field will only be present if the user is part of a team and `is_teammate` is `True`.

Return type str

class dropbox.users.**FullAccount**(*account_id=None*, *name=None*, *email=None*, *email_verified=None*, *disabled=None*, *locale=None*, *referral_link=None*, *is_paired=None*, *account_type=None*, *profile_photo_url=None*, *country=None*, *team=None*, *team_member_id=None*)

Bases: *dropbox.users.Account*

Detailed information about the current user's account.

Variables

- ***country*** – The user's two-letter country code, if available. Country codes are based on ISO 3166-1.
- ***locale*** – The language that the user specified. Locale tags will be IETF language tags.
- ***referral_link*** – The user's referral link.
- ***team*** – If this account is a member of a team, information about that team.
- ***team_member_id*** – This account's unique team member id. This field will only be present if `team` is present.
- ***is_paired*** – Whether the user has a personal and work account. If the current account is personal, then `team` will always be None, but `is_paired` will indicate if a work account is linked.
- ***account_type*** – What type of account this user has.

__init__(*account_id=None*, *name=None*, *email=None*, *email_verified=None*, *disabled=None*, *locale=None*, *referral_link=None*, *is_paired=None*, *account_type=None*, *profile_photo_url=None*, *country=None*, *team=None*, *team_member_id=None*)

account_type

What type of account this user has.

Return type *AccountType*

country

The user's two-letter country code, if available. Country codes are based on ISO 3166-1.

Return type str

is_paired

Whether the user has a personal and work account. If the current account is personal, then `team` will always be None, but `is_paired` will indicate if a work account is linked.

Return type bool

locale

The language that the user specified. Locale tags will be IETF language tags.

Return type str

referral_link
The user's referral link.

Return type str

team
If this account is a member of a team, information about that team.

Return type *FullTeam*

team_member_id
This account's unique team member id. This field will only be present if `team` is present.

Return type str

class `dropbox.users.`**`FullTeam`**(*id=None*, *name=None*, *sharing_policies=None*)
Bases: *`dropbox.users.Team`*

Detailed information about a team.

Variables ***sharing_policies*** – Team policies governing sharing.

__init__(*id=None*, *name=None*, *sharing_policies=None*)

sharing_policies
Team policies governing sharing.

Return type team_policies.TeamSharingPolicies_validator

class `dropbox.users.`**`GetAccountArg`**(*account_id=None*)
Bases: `object`

Variables ***account_id*** – A user's account identifier.

__init__(*account_id=None*)

account_id
A user's account identifier.

Return type str

class `dropbox.users.`**`GetAccountBatchArg`**(*account_ids=None*)
Bases: `object`

Variables ***account_ids*** – List of user account identifiers. Should not contain any duplicate account IDs.

__init__(*account_ids=None*)

account_ids
List of user account identifiers. Should not contain any duplicate account IDs.

Return type list of [str]

class `dropbox.users.`**`GetAccountBatchError`**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***no_account*** (*str*) – The value is an account ID specified in **:field:'GetAccountBatchArg.account_ids'** that does not exist.

get_no_account()
The value is an account ID specified in `GetAccountBatchArg.account_ids` that does not exist.

Only call this if *`is_no_account()`* is true.

Return type str

is_no_account()
Check if the union tag is `no_account`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

classmethod no_account(*val*)
Create an instance of this class set to the `no_account` tag with value `val`.

Parameters **val** (*str*) –

Return type *GetAccountBatchError*

other = GetAccountBatchError('other', None)

class dropbox.users.**GetAccountError**(*tag*, *value=None*)
Bases: `dropbox.stone_base.Union`

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables ***no_account*** – The specified `GetAccountArg.account_id` does not exist.

is_no_account()
Check if the union tag is `no_account`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

no_account = GetAccountError('no_account', None)

other = GetAccountError('other', None)

class dropbox.users.**IndividualSpaceAllocation**(*allocated=None*)
Bases: `object`

Variables ***allocated*** – The total space allocated to the user's account (bytes).

__init__(*allocated=None*)

allocated
The total space allocated to the user's account (bytes).

Return type long

class dropbox.users.**Name**(*given_name=None*, *surname=None*, *familiar_name=None*, *display_name=None*, *abbreviated_name=None*)
Bases: `object`

Representations for a person's name to assist with internationalization.

Variables

- ***given_name*** – Also known as a first name.
- ***surname*** – Also known as a last name or family name.
- ***familiar_name*** – Locale-dependent name. In the US, a person's familiar name is their `given_name`, but elsewhere, it could be any combination of a person's `given_name` and `surname`.
- ***display_name*** – A name that can be used directly to represent the name of a user's Dropbox account.
- ***abbreviated_name*** – An abbreviated form of the person's name. Their initials in most locales.

__init__(*given_name=None*, *surname=None*, *familiar_name=None*, *display_name=None*, *abbreviated_name=None*)

abbreviated_name

An abbreviated form of the person's name. Their initials in most locales.

Return type str

display_name

A name that can be used directly to represent the name of a user's Dropbox account.

Return type str

familiar_name

Locale-dependent name. In the US, a person's familiar name is their `given_name`, but elsewhere, it could be any combination of a person's `given_name` and `surname`.

Return type str

given_name

Also known as a first name.

Return type str

surname

Also known as a last name or family name.

Return type str

class `dropbox.users.`**SpaceAllocation**(*tag*, *value=None*)

Bases: `dropbox.stone_base.Union`

Space is allocated differently based on the type of account.

This class acts as a tagged union. Only one of the `is_*` methods will return true. To get the associated value of a tag (if one exists), use the corresponding `get_*` method.

Variables

- ***individual*** (`IndividualSpaceAllocation`) – The user's space allocation applies only to their individual account.
- ***team*** (`TeamSpaceAllocation`) – The user shares space with other members of their team.

get_individual()

The user's space allocation applies only to their individual account.

Only call this if *`is_individual()`* is true.

Return type *IndividualSpaceAllocation*

get_team()
The user shares space with other members of their team.

Only call this if *is_team()* is true.

Return type *TeamSpaceAllocation*

classmethod individual(*val*)
Create an instance of this class set to the `individual` tag with value `val`.

Parameters **val** (`IndividualSpaceAllocation`) –

Return type *SpaceAllocation*

is_individual()
Check if the union tag is `individual`.

Return type bool

is_other()
Check if the union tag is `other`.

Return type bool

is_team()
Check if the union tag is `team`.

Return type bool

other = SpaceAllocation('other', None)

classmethod team(*val*)
Create an instance of this class set to the `team` tag with value `val`.

Parameters **val** (`TeamSpaceAllocation`) –

Return type *SpaceAllocation*

class dropbox.users.**SpaceUsage**(*used=None*, *allocation=None*)
Bases: `object`

Information about a user's space usage and quota.

Variables

- ***used*** – The user's total space usage (bytes).
- ***allocation*** – The user's space allocation.

__init__(*used=None*, *allocation=None*)

allocation
The user's space allocation.

Return type *SpaceAllocation*

used
The user's total space usage (bytes).

Return type long

class dropbox.users.**Team**(*id=None*, *name=None*)
Bases: `object`

Information about a team.

Variables

- **id** – The team's unique ID.
- **name** – The name of the team.

__init__(*id=None*, *name=None*)

id

The team's unique ID.

Return type str

name

The name of the team.

Return type str

class dropbox.users.**TeamSpaceAllocation**(*used=None*, *allocated=None*)

Bases: object

Variables

- **used** – The total space currently used by the user's team (bytes).
- **allocated** – The total space allocated to the user's team (bytes).

__init__(*used=None*, *allocated=None*)

allocated

The total space allocated to the user's team (bytes).

Return type long

used

The total space currently used by the user's team (bytes).

Return type long

2.8 dropbox.exceptions – Exceptions

exception dropbox.exceptions.**ApiError**(*request_id*, *error*, *user_message_text*, *user_message_locale*)

Bases: dropbox.exceptions.DropboxException

Errors produced by the Dropbox API.

__init__(*request_id*, *error*, *user_message_text*, *user_message_locale*)

Parameters

- **request_id** (*(str)*) – A request_id can be shared with Dropbox Support to pinpoint the exact request that returns an error.
- **error** – An instance of the error data type for the route.
- **user_message_text** (*(str)*) – A human-readable message that can be displayed to the end user. Is None, if unavailable.
- **user_message_locale** (*(str)*) – The locale of user_message_text, if present.

exception dropbox.exceptions.**AuthError**(*request_id*, *error*)

Bases: dropbox.exceptions.HttpError

Errors due to invalid authentication credentials.

__init__(*request_id*, *error*)

exception dropbox.exceptions.**BadInputError**(*request_id*, *message*)
Bases: *dropbox.exceptions.HttpError*

Errors due to bad input parameters to an API Operation.

__init__(*request_id*, *message*)

exception dropbox.exceptions.**DropboxException**(*request_id*, **args*, ***kwargs*)
Bases: exceptions.Exception

All errors related to making an API request extend this.

__init__(*request_id*, **args*, ***kwargs*)

exception dropbox.exceptions.**HttpError**(*request_id*, *status_code*, *body*)
Bases: *dropbox.exceptions.DropboxException*

Errors produced at the HTTP layer.

__init__(*request_id*, *status_code*, *body*)

exception dropbox.exceptions.**InternalServerError**(*request_id*, *status_code*, *body*)
Bases: *dropbox.exceptions.HttpError*

Errors due to a problem on Dropbox.

exception dropbox.exceptions.**RateLimitError**(*request_id*, *error=None*, *backoff=None*)
Bases: *dropbox.exceptions.HttpError*

Error caused by rate limiting.

__init__(*request_id*, *error=None*, *backoff=None*)

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

d

dropbox.async, 46
dropbox.dropbox, 5
dropbox.exceptions, 252
dropbox.files, 48
dropbox.oauth, 3
dropbox.sharing, 97
dropbox.team, 182
dropbox.users, 245

Symbols

__init__() (dropbox.async.PollArg method), 47
__init__() (dropbox.dropbox.Dropbox method), 6
__init__() (dropbox.dropbox.DropboxTeam method), 32
__init__() (dropbox.exceptions.ApiError method), 252
__init__() (dropbox.exceptions.AuthError method), 252
__init__() (dropbox.exceptions.BadInputError method), 253
__init__() (dropbox.exceptions.DropboxException method), 253
__init__() (dropbox.exceptions.HttpError method), 253
__init__() (dropbox.exceptions.RateLimitError method), 253
__init__() (dropbox.sharing.AddFileMemberArgs method), 98
__init__() (dropbox.sharing.AddFolderMemberArg method), 100
__init__() (dropbox.sharing.AddMember method), 103
__init__() (dropbox.sharing.ChangeFileMemberAccessArgs method), 105
__init__() (dropbox.sharing.CollectionLinkMetadata method), 106
__init__() (dropbox.sharing.CreateSharedLinkArg method), 106
__init__() (dropbox.sharing.CreateSharedLinkWithSettingsArg method), 107
__init__() (dropbox.sharing.FileLinkMetadata method), 111
__init__() (dropbox.sharing.FileMemberActionResult method), 113
__init__() (dropbox.sharing.FilePermission method), 114
__init__() (dropbox.sharing.FolderLinkMetadata method), 116
__init__() (dropbox.sharing.FolderPermission method), 116
__init__() (dropbox.sharing.FolderPolicy method), 117
__init__() (dropbox.sharing.GetFileMetadataArg method), 118
__init__() (dropbox.sharing.GetFileMetadataBatchArg method), 118
__init__() (dropbox.sharing.GetFileMetadataBatchResult method), 118
__init__() (dropbox.sharing.GetMetadataArgs method), 120
__init__() (dropbox.sharing.GetSharedLinkMetadataArg method), 121
__init__() (dropbox.sharing.GetSharedLinksArg method), 121
__init__() (dropbox.sharing.GetSharedLinksResult method), 122
__init__() (dropbox.sharing.GroupInfo method), 122
__init__() (dropbox.sharing.GroupMembershipInfo method), 123
__init__() (dropbox.sharing.InsufficientQuotaAmounts method), 123
__init__() (dropbox.sharing.InviteeMembershipInfo method), 124
__init__() (dropbox.sharing.LinkMetadata method), 126
__init__() (dropbox.sharing.LinkPermissions method), 127
__init__() (dropbox.sharing.ListFileMembersArg method), 127
__init__() (dropbox.sharing.ListFileMembersBatchArg method), 128
__init__() (dropbox.sharing.ListFileMembersBatchResult method), 128
__init__() (dropbox.sharing.ListFileMembersContinueArg method), 128
__init__() (dropbox.sharing.ListFileMembersCountResult method), 130
__init__() (dropbox.sharing.ListFilesArg method), 132
__init__() (dropbox.sharing.ListFilesContinueArg method), 132
__init__() (dropbox.sharing.ListFilesResult method), 133
__init__() (dropbox.sharing.ListFolderMembersArgs method), 133
__init__() (dropbox.sharing.ListFolderMembersContinueArg method), 133
__init__() (dropbox.sharing.ListFolderMembersCursorArg method), 134

__init__() (dropbox.sharing.ListFoldersArgs method), 135
__init__() (dropbox.sharing.ListFoldersContinueArg method), 135
__init__() (dropbox.sharing.ListFoldersResult method), 136
__init__() (dropbox.sharing.ListSharedLinksArg method), 136
__init__() (dropbox.sharing.ListSharedLinksResult method), 137
__init__() (dropbox.sharing.MemberAccessLevelResult method), 138
__init__() (dropbox.sharing.MemberPermission method), 139
__init__() (dropbox.sharing.MembershipInfo method), 141
__init__() (dropbox.sharing.ModifySharedLinkSettingsArgs method), 142
__init__() (dropbox.sharing.MountFolderArg method), 143
__init__() (dropbox.sharing.ParentFolderAccessInfo method), 144
__init__() (dropbox.sharing.PathLinkMetadata method), 145
__init__() (dropbox.sharing.RelinquishFileMembershipArg method), 146
__init__() (dropbox.sharing.RelinquishFolderMembershipArg method), 147
__init__() (dropbox.sharing.RemoveFileMemberArg method), 149
__init__() (dropbox.sharing.RemoveFolderMemberArg method), 150
__init__() (dropbox.sharing.RevokeSharedLinkArg method), 154
__init__() (dropbox.sharing.ShareFolderArg method), 155
__init__() (dropbox.sharing.SharedFileMembers method), 160
__init__() (dropbox.sharing.SharedFileMetadata method), 161
__init__() (dropbox.sharing.SharedFolderMembers method), 164
__init__() (dropbox.sharing.SharedFolderMetadata method), 164
__init__() (dropbox.sharing.SharedFolderMetadataBase method), 165
__init__() (dropbox.sharing.SharedLinkMetadata method), 168
__init__() (dropbox.sharing.SharedLinkSettings method), 169
__init__() (dropbox.sharing.TeamMemberInfo method), 171
__init__() (dropbox.sharing.TransferFolderArg method), 172
__init__() (dropbox.sharing.UnmountFolderArg method), 173
__init__() (dropbox.sharing.UnshareFileArg method), 174
__init__() (dropbox.sharing.UnshareFolderArg method), 175
__init__() (dropbox.sharing.UpdateFolderMemberArg method), 176
__init__() (dropbox.sharing.UpdateFolderPolicyArg method), 178
__init__() (dropbox.sharing.UserInfo method), 180
__init__() (dropbox.sharing.UserMembershipInfo method), 180
__init__() (dropbox.team.ActiveWebSession method), 182
__init__() (dropbox.team.AddPropertyTemplateArg method), 182
__init__() (dropbox.team.AddPropertyTemplateResult method), 182
__init__() (dropbox.team.ApiApp method), 183
__init__() (dropbox.team.BaseDfbReport method), 184
__init__() (dropbox.team.DateRange method), 184
__init__() (dropbox.team.DesktopClientSession method), 185
__init__() (dropbox.team.DeviceSession method), 186
__init__() (dropbox.team.DeviceSessionArg method), 187
__init__() (dropbox.team.DevicesActive method), 187
__init__() (dropbox.team.GetActivityReport method), 189
__init__() (dropbox.team.GetDevicesReport method), 190
__init__() (dropbox.team.GetMembershipReport method), 191
__init__() (dropbox.team.GetStorageReport method), 191
__init__() (dropbox.team.GroupCreateArg method), 193
__init__() (dropbox.team.GroupFullInfo method), 194
__init__() (dropbox.team.GroupMemberInfo method), 194
__init__() (dropbox.team.GroupMemberSelector method), 195
__init__() (dropbox.team.GroupMembersAddArg method), 195
__init__() (dropbox.team.GroupMembersChangeResult method), 197
__init__() (dropbox.team.GroupMembersRemoveArg method), 198
__init__() (dropbox.team.GroupMembersSelector method), 198
__init__() (dropbox.team.GroupMembersSetAccessTypeArg method), 199
__init__() (dropbox.team.GroupUpdateArgs method), 201

__init__() (dropbox.team.GroupsListArg method), 202
__init__() (dropbox.team.GroupsListContinueArg method), 203
__init__() (dropbox.team.GroupsListResult method), 203
__init__() (dropbox.team.GroupsMembersListArg method), 204
__init__() (dropbox.team.GroupsMembersListContinueArg method), 204
__init__() (dropbox.team.GroupsMembersListResult method), 204
__init__() (dropbox.team.IncludeMembersArg method), 206
__init__() (dropbox.team.ListMemberAppsArg method), 206
__init__() (dropbox.team.ListMemberAppsResult method), 207
__init__() (dropbox.team.ListMemberDevicesArg method), 207
__init__() (dropbox.team.ListMemberDevicesResult method), 208
__init__() (dropbox.team.ListMembersAppsArg method), 208
__init__() (dropbox.team.ListMembersAppsResult method), 209
__init__() (dropbox.team.ListMembersDevicesArg method), 210
__init__() (dropbox.team.ListMembersDevicesResult method), 211
__init__() (dropbox.team.ListTeamAppsArg method), 211
__init__() (dropbox.team.ListTeamAppsResult method), 212
__init__() (dropbox.team.ListTeamDevicesArg method), 212
__init__() (dropbox.team.ListTeamDevicesResult method), 213
__init__() (dropbox.team.MemberAccess method), 214
__init__() (dropbox.team.MemberAddArg method), 214
__init__() (dropbox.team.MemberDevices method), 218
__init__() (dropbox.team.MemberLinkedApps method), 219
__init__() (dropbox.team.MemberProfile method), 219
__init__() (dropbox.team.MembersAddArg method), 220
__init__() (dropbox.team.MembersDeactivateArg method), 222
__init__() (dropbox.team.MembersGetInfoArgs method), 222
__init__() (dropbox.team.MembersListArg method), 224
__init__() (dropbox.team.MembersListContinueArg method), 224
__init__() (dropbox.team.MembersListResult method), 225
__init__() (dropbox.team.MembersRecoverArg method), 225
__init__() (dropbox.team.MembersRemoveArg method), 226
__init__() (dropbox.team.MembersSetPermissionsArg method), 229
__init__() (dropbox.team.MembersSetPermissionsResult method), 230
__init__() (dropbox.team.MembersSetProfileArg method), 230
__init__() (dropbox.team.MembersUnsuspendArg method), 232
__init__() (dropbox.team.MobileClientSession method), 234
__init__() (dropbox.team.RemovedStatus method), 234
__init__() (dropbox.team.RevokeDesktopClientArg method), 235
__init__() (dropbox.team.RevokeDeviceSessionBatchArg method), 236
__init__() (dropbox.team.RevokeDeviceSessionBatchResult method), 236
__init__() (dropbox.team.RevokeDeviceSessionStatus method), 237
__init__() (dropbox.team.RevokeLinkedApiAppArg method), 237
__init__() (dropbox.team.RevokeLinkedApiAppBatchArg method), 238
__init__() (dropbox.team.RevokeLinkedAppBatchResult method), 238
__init__() (dropbox.team.RevokeLinkedAppStatus method), 239
__init__() (dropbox.team.StorageBucket method), 239
__init__() (dropbox.team.TeamGetInfoResult method), 239
__init__() (dropbox.team.TeamMemberInfo method), 240
__init__() (dropbox.team.TeamMemberProfile method), 240
__init__() (dropbox.team.UpdatePropertyTemplateArg method), 242
__init__() (dropbox.team.UpdatePropertyTemplateResult method), 242
__init__() (dropbox.users.Account method), 245
__init__() (dropbox.users.BasicAccount method), 246
__init__() (dropbox.users.FullAccount method), 247
__init__() (dropbox.users.FullTeam method), 248
__init__() (dropbox.users.GetAccountArg method), 248
__init__() (dropbox.users.GetAccountBatchArg method), 248
__init__() (dropbox.users.IndividualSpaceAllocation method), 249
__init__() (dropbox.users.Name method), 250
__init__() (dropbox.users.SpaceUsage method), 251
__init__() (dropbox.users.Team method), 252
__init__() (dropbox.users.TeamSpaceAllocation method), 252

A

abbreviated_name (dropbox.users.Name attribute), 250
access_denied (dropbox.sharing.CreateSharedLinkWithSettingsError attribute), 107
access_denied (dropbox.team.GroupsPollError attribute), 205
access_details (dropbox.sharing.MemberAccessLevelResult attribute), 138
access_error() (dropbox.sharing.AddFileMemberError class method), 99
access_error() (dropbox.sharing.AddFolderMemberError class method), 101
access_error() (dropbox.sharing.FileMemberActionError class method), 111
access_error() (dropbox.sharing.GetFileMetadataError class method), 118
access_error() (dropbox.sharing.GetFileMetadataIndividualResult class method), 119
access_error() (dropbox.sharing.ListFileMembersContinueError class method), 129
access_error() (dropbox.sharing.ListFileMembersError class method), 130
access_error() (dropbox.sharing.ListFileMembersIndividualResult class method), 131
access_error() (dropbox.sharing.ListFolderMembersContinueError class method), 133
access_error() (dropbox.sharing.MountFolderError class method), 143
access_error() (dropbox.sharing.RelinquishFileMembershipError class method), 147
access_error() (dropbox.sharing.RelinquishFolderMembershipError class method), 148
access_error() (dropbox.sharing.RemoveFileMemberError class method), 149
access_error() (dropbox.sharing.RemoveFolderMemberError class method), 151
access_error() (dropbox.sharing.TransferFolderError class method), 172
access_error() (dropbox.sharing.UnmountFolderError class method), 174
access_error() (dropbox.sharing.UnshareFileError class method), 174
access_error() (dropbox.sharing.UnshareFolderError class method), 176
access_error() (dropbox.sharing.UpdateFolderMemberError class method), 177
access_error() (dropbox.sharing.UpdateFolderPolicyError class method), 179
access_level (dropbox.sharing.AddFileMemberArgs attribute), 99
access_level (dropbox.sharing.AddMember attribute), 103
access_level (dropbox.sharing.ChangeFileMemberAccessArgs attribute), 105
access_level (dropbox.sharing.MemberAccessLevelResult attribute), 138
access_level (dropbox.sharing.UpdateFolderMemberArg attribute), 176
access_type (dropbox.sharing.MembershipInfo attribute), 141
access_type (dropbox.sharing.SharedFolderMetadataBase attribute), 165
access_type (dropbox.team.GroupMemberInfo attribute), 194
access_type (dropbox.team.GroupMembersSetAccessTypeArg attribute), 199
access_type (dropbox.team.MemberAccess attribute), 214
AccessLevel (class in dropbox.sharing), 97
Account (class in dropbox.users), 245
account_id (dropbox.sharing.UserInfo attribute), 180
account_id (dropbox.team.MemberProfile attribute), 219
account_id (dropbox.users.Account attribute), 245
account_id (dropbox.users.GetAccountArg attribute), 248
account_ids (dropbox.users.GetAccountBatchArg attribute), 248
account_type (dropbox.users.FullAccount attribute), 247
AccountType (class in dropbox.users), 246
acl_update_policy (dropbox.sharing.FolderPolicy attribute), 117
acl_update_policy (dropbox.sharing.ShareFolderArg attribute), 155
acl_update_policy (dropbox.sharing.UpdateFolderPolicyArg attribute), 178
AclUpdatePolicy (class in dropbox.sharing), 98
action (dropbox.sharing.FilePermission attribute), 114
action (dropbox.sharing.FolderPermission attribute), 116
action (dropbox.sharing.MemberPermission attribute), 139
actions (dropbox.sharing.GetFileMetadataArg attribute), 118
actions (dropbox.sharing.GetFileMetadataBatchArg attribute), 118
actions (dropbox.sharing.GetMetadataArgs attribute), 120
actions (dropbox.sharing.ListFileMembersArg attribute), 127
actions (dropbox.sharing.ListFilesArg attribute), 132
actions (dropbox.sharing.ListFolderMembersCursorArg attribute), 134
actions (dropbox.sharing.ListFoldersArgs attribute), 135
active (dropbox.team.TeamMemberStatus attribute), 241
active_1_day (dropbox.team.GetDevicesReport attribute), 190
active_28_day (dropbox.team.GetDevicesReport attribute), 190

active_7_day (dropbox.team.GetDevicesReport attribute), 190
active_shared_folders_1_day (dropbox.team.GetActivityReport attribute), 189
active_shared_folders_28_day (dropbox.team.GetActivityReport attribute), 189
active_shared_folders_7_day (dropbox.team.GetActivityReport attribute), 189
active_users_1_day (dropbox.team.GetActivityReport attribute), 189
active_users_28_day (dropbox.team.GetActivityReport attribute), 189
active_users_7_day (dropbox.team.GetActivityReport attribute), 189
active_web_sessions (dropbox.team.ListMemberDevicesResult attribute), 208
ActiveWebSession (class in dropbox.team), 182
add (dropbox.files.WriteMode attribute), 96
add_fields (dropbox.team.UpdatePropertyTemplateArg attribute), 242
add_message_as_comment (dropbox.sharing.AddFileMemberArgs attribute), 99
add_or_update_fields (dropbox.files.PropertyGroupUpdate attribute), 71
AddFileMemberArgs (class in dropbox.sharing), 98
AddFileMemberError (class in dropbox.sharing), 99
AddFolderMemberArg (class in dropbox.sharing), 100
AddFolderMemberError (class in dropbox.sharing), 101
AddMember (class in dropbox.sharing), 103
AddMemberSelectorError (class in dropbox.sharing), 103
AddPropertiesError (class in dropbox.files), 48
AddPropertyTemplateArg (class in dropbox.team), 182
AddPropertyTemplateResult (class in dropbox.team), 182
adds (dropbox.team.GetActivityReport attribute), 189
AdminTier (class in dropbox.team), 182
allocated (dropbox.users.IndividualSpaceAllocation attribute), 249
allocated (dropbox.users.TeamSpaceAllocation attribute), 252
allocation (dropbox.users.SpaceUsage attribute), 251
allow (dropbox.sharing.FilePermission attribute), 114
allow (dropbox.sharing.FolderPermission attribute), 116
allow (dropbox.sharing.MemberPermission attribute), 139
allow_shared_folder (dropbox.files.RelocationArg attribute), 72
allow_shared_folder (dropbox.files.RelocationBatchArg attribute), 72
AlphaGetMetadataArg (class in dropbox.files), 48
AlphaGetMetadataError (class in dropbox.files), 49
already_mounted (dropbox.sharing.MountFolderError attribute), 143
already_shared() (dropbox.sharing.SharePathError class method), 158
android (dropbox.team.DevicesActive attribute), 187
android (dropbox.team.MobileClientPlatform attribute), 233
anyone (dropbox.sharing.MemberPolicy attribute), 140
anyone (dropbox.sharing.SharedLinkPolicy attribute), 168
ApiApp (class in dropbox.team), 183
ApiError, 252
app_id (dropbox.team.ApiApp attribute), 183
app_id (dropbox.team.RevokeLinkedApiAppArg attribute), 237
app_name (dropbox.team.ApiApp attribute), 183
app_not_found (dropbox.team.RevokeLinkedAppError attribute), 238
apps (dropbox.team.ListMembersAppsResult attribute), 209
apps (dropbox.team.ListTeamAppsResult attribute), 212
as_user() (dropbox.dropbox.DropboxTeam method), 33
async_job_id (dropbox.async.PollArg attribute), 47
async_job_id (dropbox.team.GroupMembersChangeResult attribute), 197
async_job_id() (dropbox.async.LaunchResultBase class method), 47
auth_token_revoke() (dropbox.dropbox.Dropbox method), 6
AuthError, 252
automatic_group (dropbox.sharing.AddMemberSelectorError attribute), 104
autorename (dropbox.files.CommitInfo attribute), 49
autorename (dropbox.files.CreateFolderArg attribute), 50
autorename (dropbox.files.RelocationArg attribute), 72
autorename (dropbox.files.RelocationBatchArg attribute), 73

B

backoff (dropbox.files.ListFolderLongpollResult attribute), 64
bad_member() (dropbox.sharing.AddFolderMemberError class method), 101
bad_path() (dropbox.sharing.ShareFolderErrorBase class method), 156
BadInputError, 253
BadRequestException, 3
BadStateException, 3
BaseDfbReport (class in dropbox.team), 184
basic (dropbox.users.AccountType attribute), 246
BasicAccount (class in dropbox.users), 246
blackberry (dropbox.team.MobileClientPlatform attribute), 233

both (dropbox.files.SearchMatchType attribute), 83
browser (dropbox.team.ActiveWebSession attribute), 182
bucket (dropbox.team.StorageBucket attribute), 239
business (dropbox.users.AccountType attribute), 246

C

can_revoke (dropbox.sharing.LinkPermissions attribute), 127
cannot_keep_account_and_delete_data (dropbox.team.MembersRemoveError attribute), 227
cannot_keep_account_and_transfer (dropbox.team.MembersRemoveError attribute), 227
cannot_set_permissions (dropbox.team.MembersSetPermissionsError attribute), 229
cant_copy_shared_folder (dropbox.files.RelocationError attribute), 74
cant_move_folder_into_itself (dropbox.files.RelocationError attribute), 74
cant_nest_shared_folder (dropbox.files.RelocationError attribute), 74
cant_share_outside_team (dropbox.sharing.AddFolderMemberError attribute), 101
change_options (dropbox.sharing.FolderAction attribute), 115
ChangeFileMemberAccessArgs (class in dropbox.sharing), 105
changes (dropbox.files.ListFolderLongpollResult attribute), 64
client_modified (dropbox.files.CommitInfo attribute), 49
client_modified (dropbox.files.FileMetadata attribute), 55
client_modified (dropbox.sharing.FileLinkMetadata attribute), 111
client_type (dropbox.team.DesktopClientSession attribute), 185
client_type (dropbox.team.MobileClientSession attribute), 234
client_version (dropbox.team.DesktopClientSession attribute), 185
client_version (dropbox.team.MobileClientSession attribute), 234
close (dropbox.files.UploadSessionAppendArg attribute), 89
close (dropbox.files.UploadSessionStartArg attribute), 93
closed (dropbox.files.UploadSessionLookupError attribute), 92
CollectionLinkMetadata (class in dropbox.sharing), 105
commit (dropbox.files.UploadSessionFinishArg attribute), 89
CommitInfo (class in dropbox.files), 49
CommitInfoWithProperties (class in dropbox.files), 50
complete (dropbox.async.LaunchEmptyResult attribute), 46
complete (dropbox.async.PollEmptyResult attribute), 47
complete (dropbox.sharing.JobStatus attribute), 125
complete() (dropbox.files.DeleteBatchJobStatus class method), 51
complete() (dropbox.files.RelocationBatchJobStatus class method), 73
complete() (dropbox.files.SaveUrlJobStatus class method), 81
complete() (dropbox.files.SaveUrlResult class method), 81
complete() (dropbox.files.UploadSessionFinishBatchJobStatus class method), 90
complete() (dropbox.sharing.RemoveMemberJobStatus class method), 152
complete() (dropbox.sharing.ShareFolderJobStatus class method), 156
complete() (dropbox.sharing.ShareFolderLaunch class method), 157
complete() (dropbox.team.MembersAddJobStatus class method), 221
complete() (dropbox.team.MembersAddLaunch class method), 221
conflict() (dropbox.files.WriteError class method), 95
contains_app_folder (dropbox.sharing.SharePathError attribute), 158
contains_shared_folder (dropbox.sharing.SharePathError attribute), 158
contains_team_folder (dropbox.sharing.SharePathError attribute), 158
content (dropbox.files.SearchMatchType attribute), 83
content_owner_team_info (dropbox.sharing.SharedLinkMetadata attribute), 168
conversion_error (dropbox.files.ThumbnailError attribute), 85
copy_reference (dropbox.files.GetCopyReferenceResult attribute), 59
copy_reference (dropbox.files.SaveCopyReferenceArg attribute), 78
correct_offset (dropbox.files.UploadSessionOffsetError attribute), 93
country (dropbox.team.DeviceSession attribute), 186
country (dropbox.users.FullAccount attribute), 247
create_link (dropbox.sharing.FileAction attribute), 108
create_link (dropbox.sharing.FolderAction attribute), 115
create_session() (in module dropbox.dropbox), 46
created (dropbox.team.DeviceSession attribute), 186
created (dropbox.team.GroupFullInfo attribute), 194
CreateFolderArg (class in dropbox.files), 50
CreateFolderError (class in dropbox.files), 50
CreateSharedLinkArg (class in dropbox.sharing), 106
CreateSharedLinkError (class in dropbox.sharing), 106

CreateSharedLinkWithSettingsArg (class in dropbox.sharing), 107
CreateSharedLinkWithSettingsError (class in dropbox.sharing), 107
CsrfException, 3
cursor (dropbox.files.ListFolderContinueArg attribute), 62
cursor (dropbox.files.ListFolderGetLatestCursorResult attribute), 63
cursor (dropbox.files.ListFolderLongpollArg attribute), 64
cursor (dropbox.files.ListFolderResult attribute), 65
cursor (dropbox.files.UploadSessionAppendArg attribute), 89
cursor (dropbox.files.UploadSessionFinishArg attribute), 89
cursor (dropbox.sharing.ListFileMembersContinueArg attribute), 128
cursor (dropbox.sharing.ListFilesContinueArg attribute), 132
cursor (dropbox.sharing.ListFilesResult attribute), 133
cursor (dropbox.sharing.ListFolderMembersContinueArg attribute), 133
cursor (dropbox.sharing.ListFoldersContinueArg attribute), 135
cursor (dropbox.sharing.ListFoldersResult attribute), 136
cursor (dropbox.sharing.ListSharedLinksArg attribute), 136
cursor (dropbox.sharing.ListSharedLinksResult attribute), 137
cursor (dropbox.sharing.SharedFileMembers attribute), 160
cursor (dropbox.sharing.SharedFolderMembers attribute), 164
cursor (dropbox.team.GroupsListContinueArg attribute), 203
cursor (dropbox.team.GroupsListResult attribute), 203
cursor (dropbox.team.GroupsMembersListContinueArg attribute), 204
cursor (dropbox.team.GroupsMembersListResult attribute), 204
cursor (dropbox.team.ListMembersAppsArg attribute), 208
cursor (dropbox.team.ListMembersAppsResult attribute), 209
cursor (dropbox.team.ListMembersDevicesArg attribute), 210
cursor (dropbox.team.ListMembersDevicesResult attribute), 211
cursor (dropbox.team.ListTeamAppsArg attribute), 211
cursor (dropbox.team.ListTeamAppsResult attribute), 212
cursor (dropbox.team.ListTeamDevicesArg attribute), 212
cursor (dropbox.team.ListTeamDevicesResult attribute), 213
cursor (dropbox.team.MembersListContinueArg attribute), 224
cursor (dropbox.team.MembersListResult attribute), 225
custom_message (dropbox.sharing.AddFileMemberArgs attribute), 99
custom_message (dropbox.sharing.AddFolderMemberArg attribute), 100

D

DateRange (class in dropbox.team), 184
DateRangeError (class in dropbox.team), 184
delete_on_unlink (dropbox.team.RevokeDesktopClientArg attribute), 235
DeleteArg (class in dropbox.files), 51
DeleteBatchArg (class in dropbox.files), 51
DeleteBatchError (class in dropbox.files), 51
DeleteBatchJobStatus (class in dropbox.files), 51
DeleteBatchResult (class in dropbox.files), 52
DeleteBatchResultEntry (class in dropbox.files), 52
deleted_filename (dropbox.files.SearchMode attribute), 84
DeletedMetadata (class in dropbox.files), 54
DeleteError (class in dropbox.files), 53
DeleteResult (class in dropbox.files), 54
deletes (dropbox.team.GetActivityReport attribute), 189
description (dropbox.team.UpdatePropertyTemplateArg attribute), 242
desktop_client() (dropbox.team.RevokeDeviceSessionArg class method), 235
desktop_client_sessions (dropbox.team.ListMemberDevicesResult attribute), 208
desktop_clients (dropbox.team.MemberDevices attribute), 218
DesktopClientSession (class in dropbox.team), 185
DesktopPlatform (class in dropbox.team), 185
device_name (dropbox.team.MobileClientSession attribute), 234
device_session_not_found (dropbox.team.RevokeDeviceSessionError attribute), 236
devices (dropbox.team.ListMembersDevicesResult attribute), 211
devices (dropbox.team.ListTeamDevicesResult attribute), 213
DevicesActive (class in dropbox.team), 187
DeviceSession (class in dropbox.team), 186
DeviceSessionArg (class in dropbox.team), 187
Dimensions (class in dropbox.files), 54
dimensions (dropbox.files.MediaMetadata attribute), 68

direct_only (dropbox.sharing.ListSharedLinksArg attribute), 136
disabled (dropbox.users.Account attribute), 245
disallowed_name (dropbox.files.WriteError attribute), 95
disallowed_shared_link_policy (dropbox.sharing.ShareFolderErrorBase attribute), 156
disallowed_shared_link_policy (dropbox.sharing.UpdateFolderPolicyError attribute), 179
display_name (dropbox.sharing.TeamMemberInfo attribute), 171
display_name (dropbox.users.Name attribute), 250
does_not_fit_template (dropbox.files.InvalidPropertyGroupError attribute), 61
download_failed (dropbox.files.SaveUrlError attribute), 80
DownloadArg (class in dropbox.files), 54
DownloadError (class in dropbox.files), 54
Dropbox (class in dropbox.dropbox), 5
dropbox.async (module), 46
dropbox.dropbox (module), 5
dropbox.exceptions (module), 252
dropbox.files (module), 48
dropbox.oauth (module), 3
dropbox.sharing (module), 97
dropbox.team (module), 182
dropbox.users (module), 245
dropbox_id() (dropbox.sharing.MemberSelector class method), 140
DropboxException, 253
DropboxOAuth2Flow (class in dropbox.oauth), 3
DropboxOAuth2FlowNoRedirect (class in dropbox.oauth), 4
DropboxTeam (class in dropbox.dropbox), 32
duplicate_external_member_id() (dropbox.team.MemberAddResult class method), 215
duplicate_user (dropbox.team.GroupMembersAddError attribute), 196
duplicated_or_nested_paths (dropbox.files.RelocationBatchError attribute), 73
duration (dropbox.files.VideoMetadata attribute), 94

E

edit_contents (dropbox.sharing.FileAction attribute), 108
edit_contents (dropbox.sharing.FolderAction attribute), 115
editor (dropbox.sharing.AccessLevel attribute), 97
editors (dropbox.sharing.AclUpdatePolicy attribute), 98
edits (dropbox.team.GetActivityReport attribute), 189
email (dropbox.team.MemberProfile attribute), 219
email (dropbox.users.Account attribute), 245
email() (dropbox.sharing.InviteeInfo class method), 123
email() (dropbox.sharing.MemberSelector class method), 140
email() (dropbox.team.UserSelectorArg class method), 243
email_address_too_long_to_be_disabled (dropbox.team.MembersRemoveError attribute), 227
email_not_verified (dropbox.sharing.CreateSharedLinkWithSettingsError attribute), 107
email_not_verified (dropbox.sharing.ModifySharedLinkSettingsError attribute), 142
email_reserved_for_other_user (dropbox.team.MembersSetProfileError attribute), 231
email_unverified (dropbox.sharing.AddFolderMemberError attribute), 101
email_unverified (dropbox.sharing.SharedFolderAccessError attribute), 162
email_unverified (dropbox.sharing.ShareFolderErrorBase attribute), 156
email_unverified (dropbox.sharing.SharingUserError attribute), 171
email_verified (dropbox.team.MemberProfile attribute), 219
email_verified (dropbox.users.Account attribute), 245
email_verify_required (dropbox.sharing.SharedLinkAccessFailureReason attribute), 166
emails() (dropbox.team.UsersSelectorArg class method), 244
end_date (dropbox.team.DateRange attribute), 184
entries (dropbox.files.DeleteBatchArg attribute), 51
entries (dropbox.files.DeleteBatchResult attribute), 52
entries (dropbox.files.ListFolderResult attribute), 65
entries (dropbox.files.ListRevisionsResult attribute), 66
entries (dropbox.files.RelocationBatchArg attribute), 73
entries (dropbox.files.RelocationBatchResult attribute), 74
entries (dropbox.files.UploadSessionFinishBatchArg attribute), 90
entries (dropbox.files.UploadSessionFinishBatchResult attribute), 90
entries (dropbox.sharing.ListFilesResult attribute), 133
entries (dropbox.sharing.ListFoldersResult attribute), 136
error_type (dropbox.team.RevokeDeviceSessionStatus attribute), 237
error_type (dropbox.team.RevokeLinkedAppStatus attribute), 239

expires (dropbox.files.GetCopyReferenceResult attribute), 59
expires (dropbox.sharing.LinkMetadata attribute), 126
expires (dropbox.sharing.SharedLinkMetadata attribute), 168
expires (dropbox.sharing.SharedLinkSettings attribute), 169
external_id (dropbox.team.MemberProfile attribute), 219
external_id() (dropbox.team.UserSelectorArg class method), 243
external_id_already_in_use (dropbox.team.GroupCreateError attribute), 193
external_id_already_in_use (dropbox.team.GroupUpdateError attribute), 201
external_id_and_new_external_id_unsafe (dropbox.team.MembersSetProfileError attribute), 231
external_id_used_by_other_user (dropbox.team.MembersSetProfileError attribute), 231
external_ids() (dropbox.team.UsersSelectorArg class method), 244

F

failed() (dropbox.files.DeleteBatchJobStatus class method), 52
failed() (dropbox.files.RelocationBatchJobStatus class method), 73
failed() (dropbox.files.SaveUrlJobStatus class method), 81
failed() (dropbox.sharing.JobStatus class method), 125
failed() (dropbox.sharing.RemoveMemberJobStatus class method), 152
failed() (dropbox.sharing.ShareFolderJobStatus class method), 156
failed() (dropbox.team.MembersAddJobStatus class method), 221
failure() (dropbox.files.DeleteBatchResultEntry class method), 52
failure() (dropbox.files.UploadSessionFinishBatchResultEntry class method), 90
familiar_name (dropbox.users.Name attribute), 250
file (dropbox.files.WriteConflictError attribute), 94
file (dropbox.sharing.AddFileMemberArgs attribute), 99
file (dropbox.sharing.ChangeFileMemberAccessArgs attribute), 105
file (dropbox.sharing.GetFileMetadataArg attribute), 118
file (dropbox.sharing.GetFileMetadataBatchResult attribute), 118
file (dropbox.sharing.ListFileMembersArg attribute), 127
file (dropbox.sharing.ListFileMembersBatchResult attribute), 128
file (dropbox.sharing.PendingUploadMode attribute), 145
file (dropbox.sharing.RelinquishFileMembershipArg attribute), 146
file (dropbox.sharing.RemoveFileMemberArg attribute), 149
file (dropbox.sharing.UnshareFileArg attribute), 174
file_ancestor (dropbox.files.WriteConflictError attribute), 94
file_not_found_error() (dropbox.sharing.FileErrorResult class method), 109
FileAction (class in dropbox.sharing), 108
FileErrorResult (class in dropbox.sharing), 109
FileLinkMetadata (class in dropbox.sharing), 110
FileMemberActionError (class in dropbox.sharing), 111
FileMemberActionIndividualResult (class in dropbox.sharing), 112
FileMemberActionResult (class in dropbox.sharing), 113
FileMemberRemoveActionResult (class in dropbox.sharing), 113
FileMetadata (class in dropbox.files), 55
filename (dropbox.files.SearchMatchType attribute), 83
filename (dropbox.files.SearchMode attribute), 84
filename_and_content (dropbox.files.SearchMode attribute), 84
FilePermission (class in dropbox.sharing), 114
files (dropbox.sharing.GetFileMetadataBatchArg attribute), 118
files (dropbox.sharing.ListFileMembersBatchArg attribute), 128
files_alpha_get_metadata() (dropbox.dropbox.Dropbox method), 6
files_alpha_upload() (dropbox.dropbox.Dropbox method), 6
files_copy() (dropbox.dropbox.Dropbox method), 7
files_copy_batch() (dropbox.dropbox.Dropbox method), 7
files_copy_batch_check() (dropbox.dropbox.Dropbox method), 7
files_copy_reference_get() (dropbox.dropbox.Dropbox method), 7
files_copy_reference_save() (dropbox.dropbox.Dropbox method), 8
files_create_folder() (dropbox.dropbox.Dropbox method), 8
files_delete() (dropbox.dropbox.Dropbox method), 8
files_delete_batch() (dropbox.dropbox.Dropbox method), 8
files_delete_batch_check() (dropbox.dropbox.Dropbox method), 9
files_download() (dropbox.dropbox.Dropbox method), 9
files_download_to_file() (dropbox.dropbox.Dropbox method), 9
files_get_metadata() (dropbox.dropbox.Dropbox method), 9

files_get_preview() (dropbox.dropbox.Dropbox method), 10
files_get_preview_to_file() (dropbox.dropbox.Dropbox method), 10
files_get_temporary_link() (dropbox.dropbox.Dropbox method), 10
files_get_thumbnail() (dropbox.dropbox.Dropbox method), 10
files_get_thumbnail_to_file() (dropbox.dropbox.Dropbox method), 11
files_list_folder() (dropbox.dropbox.Dropbox method), 11
files_list_folder_continue() (dropbox.dropbox.Dropbox method), 12
files_list_folder_get_latest_cursor() (dropbox.dropbox.Dropbox method), 12
files_list_folder_longpoll() (dropbox.dropbox.Dropbox method), 13
files_list_revisions() (dropbox.dropbox.Dropbox method), 13
files_move() (dropbox.dropbox.Dropbox method), 13
files_move_batch() (dropbox.dropbox.Dropbox method), 14
files_move_batch_check() (dropbox.dropbox.Dropbox method), 14
files_permanently_delete() (dropbox.dropbox.Dropbox method), 14
files_properties_add() (dropbox.dropbox.Dropbox method), 14
files_properties_overwrite() (dropbox.dropbox.Dropbox method), 15
files_properties_remove() (dropbox.dropbox.Dropbox method), 15
files_properties_template_get() (dropbox.dropbox.Dropbox method), 15
files_properties_template_list() (dropbox.dropbox.Dropbox method), 15
files_properties_update() (dropbox.dropbox.Dropbox method), 15
files_restore() (dropbox.dropbox.Dropbox method), 16
files_save_url() (dropbox.dropbox.Dropbox method), 16
files_save_url_check_job_status() (dropbox.dropbox.Dropbox method), 16
files_search() (dropbox.dropbox.Dropbox method), 16
files_upload() (dropbox.dropbox.Dropbox method), 17
files_upload_session_append() (dropbox.dropbox.Dropbox method), 17
files_upload_session_append_v2() (dropbox.dropbox.Dropbox method), 18
files_upload_session_finish() (dropbox.dropbox.Dropbox method), 18
files_upload_session_finish_batch() (dropbox.dropbox.Dropbox method), 18
files_upload_session_finish_batch_check() (dropbox.dropbox.Dropbox method), 18
files_upload_session_start() (dropbox.dropbox.Dropbox method), 19
FileSharingInfo (class in dropbox.files), 56
finish() (dropbox.oauth.DropboxOAuth2Flow method), 4
finish() (dropbox.oauth.DropboxOAuth2FlowNoRedirect method), 5
folder (dropbox.files.WriteConflictError attribute), 94
folder (dropbox.sharing.PendingUploadMode attribute), 145
folder_is_limited_team_folder (dropbox.sharing.PermissionDeniedReason attribute), 145
folder_name (dropbox.sharing.ParentFolderAccessInfo attribute), 144
folder_owner (dropbox.sharing.RelinquishFolderMembershipError attribute), 148
folder_owner (dropbox.sharing.RemoveFolderMemberError attribute), 151
FolderAction (class in dropbox.sharing), 114
FolderLinkMetadata (class in dropbox.sharing), 116
FolderMetadata (class in dropbox.files), 57
FolderPermission (class in dropbox.sharing), 116
FolderPolicy (class in dropbox.sharing), 117
FolderSharingInfo (class in dropbox.files), 57
force_async (dropbox.sharing.ShareFolderArg attribute), 155
force_async (dropbox.team.MembersAddArg attribute), 220
format (dropbox.files.ThumbnailArg attribute), 85
free_team_member_limit_reached() (dropbox.team.MemberAddResult class method), 215
from_lookup() (dropbox.files.RelocationError class method), 74
from_no_redirect_result() (dropbox.oauth.OAuth2FlowResult class method), 5
from_path (dropbox.files.RelocationPath attribute), 76
from_write() (dropbox.files.RelocationError class method), 74
full (dropbox.team.TeamMembershipType attribute), 242
FullAccount (class in dropbox.users), 247
FullTeam (class in dropbox.users), 248

G

get_access_error() (dropbox.sharing.AddFileMemberError method), 99
get_access_error() (dropbox.sharing.AddFolderMemberError method), 101

get_access_error() (dropbox.sharing.FileMemberActionError method), 111
get_access_error() (dropbox.sharing.GetFileMetadataError method), 119
get_access_error() (dropbox.sharing.GetFileMetadataIndividualResult method), 119
get_access_error() (dropbox.sharing.ListFileMembersContinueError method), 129
get_access_error() (dropbox.sharing.ListFileMembersError method), 130
get_access_error() (dropbox.sharing.ListFileMembersIndividualResult method), 131
get_access_error() (dropbox.sharing.ListFolderMembersContinueError method), 134
get_access_error() (dropbox.sharing.MountFolderError method), 143
get_access_error() (dropbox.sharing.RelinquishFileMembershipError method), 147
get_access_error() (dropbox.sharing.RelinquishFolderMembershipError method), 148
get_access_error() (dropbox.sharing.RemoveFileMemberError method), 149
get_access_error() (dropbox.sharing.RemoveFolderMemberError method), 151
get_access_error() (dropbox.sharing.TransferFolderError method), 172
get_access_error() (dropbox.sharing.UnmountFolderError method), 174
get_access_error() (dropbox.sharing.UnshareFileError method), 175
get_access_error() (dropbox.sharing.UnshareFolderError method), 176
get_access_error() (dropbox.sharing.UpdateFolderMemberError method), 177
get_access_error() (dropbox.sharing.UpdateFolderPolicyError method), 179
get_already_shared() (dropbox.sharing.SharePathError method), 158
get_async_job_id() (dropbox.async.LaunchResultBase method), 47
get_bad_member() (dropbox.sharing.AddFolderMemberError method), 101
get_bad_path() (dropbox.sharing.ShareFolderErrorBase method), 156
get_complete() (dropbox.files.DeleteBatchJobStatus method), 52
get_complete() (dropbox.files.RelocationBatchJobStatus method), 74
get_complete() (dropbox.files.SaveUrlJobStatus method), 81
get_complete() (dropbox.files.SaveUrlResult method), 81
get_complete() (dropbox.files.UploadSessionFinishBatchJobStatus method), 90
get_complete() (dropbox.sharing.RemoveMemberJobStatus method), 152
get_complete() (dropbox.sharing.ShareFolderJobStatus method), 157
get_complete() (dropbox.sharing.ShareFolderLaunch method), 157
get_complete() (dropbox.team.MembersAddJobStatus method), 221
get_complete() (dropbox.team.MembersAddLaunch method), 221
get_conflict() (dropbox.files.WriteError method), 95
get_desktop_client() (dropbox.team.RevokeDeviceSessionArg method), 235
get_dropbox_id() (dropbox.sharing.MemberSelector method), 140
get_duplicate_external_member_id() (dropbox.team.MemberAddResult method), 215
get_email() (dropbox.sharing.InviteeInfo method), 123
get_email() (dropbox.sharing.MemberSelector method), 141
get_email() (dropbox.team.UserSelectorArg method), 243
get_emails() (dropbox.team.UsersSelectorArg method), 244
get_external_id() (dropbox.team.UserSelectorArg method), 243
get_external_ids() (dropbox.team.UsersSelectorArg method), 244
get_failed() (dropbox.files.DeleteBatchJobStatus method), 52
get_failed() (dropbox.files.RelocationBatchJobStatus method), 74
get_failed() (dropbox.files.SaveUrlJobStatus method), 81
get_failed() (dropbox.sharing.JobStatus method), 126
get_failed() (dropbox.sharing.RemoveMemberJobStatus method), 152
get_failed() (dropbox.sharing.ShareFolderJobStatus method), 157

get_failed() (dropbox.team.MembersAddJobStatus method), 221
get_failure() (dropbox.files.DeleteBatchResultEntry method), 52
get_failure() (dropbox.files.UploadSessionFinishBatchResultEntry method), 90
get_file_not_found_error() (dropbox.sharing.FileErrorResult method), 109
get_free_team_member_limit_reached() (dropbox.team.MemberAddResult method), 216
get_from_lookup() (dropbox.files.RelocationError method), 75
get_from_write() (dropbox.files.RelocationError method), 75
get_group_external_id() (dropbox.team.GroupSelector method), 199
get_group_external_ids() (dropbox.team.GroupsSelector method), 205
get_group_id() (dropbox.team.GroupSelector method), 199
get_group_ids() (dropbox.team.GroupsSelector method), 205
get_group_info() (dropbox.team.GroupsGetInfoItem method), 202
get_id_not_found() (dropbox.team.GroupsGetInfoItem method), 202
get_id_not_found() (dropbox.team.MembersGetInfoItem method), 223
get_incorrect_offset() (dropbox.files.UploadSessionLookupError method), 92
get_individual() (dropbox.users.SpaceAllocation method), 250
get_insufficient_quota() (dropbox.sharing.MountFolderError method), 143
get_invalid_dropbox_id() (dropbox.sharing.AddMemberSelectorError method), 104
get_invalid_email() (dropbox.sharing.AddMemberSelectorError method), 104
get_invalid_file_action_error() (dropbox.sharing.FileErrorResult method), 110
get_invalid_path_root() (dropbox.files.LookupError method), 67
get_invalid_path_root() (dropbox.sharing.SharePathError method), 158
get_lookup_failed() (dropbox.files.UploadSessionFinishError method), 91
get_malformed_path() (dropbox.files.LookupError method), 67
get_malformed_path() (dropbox.files.WriteError method), 95
get_member_error() (dropbox.sharing.FileMemberActionIndividualResult method), 112
get_member_error() (dropbox.sharing.FileMemberRemoveActionResult method), 113
get_member_error() (dropbox.sharing.RemoveFolderMemberError method), 151
get_member_error() (dropbox.sharing.UpdateFolderMemberError method), 177
get_member_info() (dropbox.team.MembersGetInfoItem method), 223
get_members_not_in_team() (dropbox.team.GroupMembersAddError method), 196
get_metadata() (dropbox.files.MediaInfo method), 68
get_metadata() (dropbox.sharing.GetFileMetadataIndividualResult method), 119
get_mobile_client() (dropbox.team.RevokeDeviceSessionArg method), 235
get_no_account() (dropbox.users.GetAccountBatchError method), 248
get_no_explicit_access() (dropbox.sharing.RemoveFileMemberError method), 149
get_no_explicit_access() (dropbox.sharing.SharedFolderMemberError method), 163
get_no_explicit_access() (dropbox.sharing.UpdateFolderMemberError method), 177
get_path() (dropbox.files.CreateFolderError method), 50
get_path() (dropbox.files.DownloadError method), 54
get_path() (dropbox.files.GetCopyReferenceError method), 58
get_path() (dropbox.files.GetMetadataError method), 59
get_path() (dropbox.files.GetTemporaryLinkError method), 60
get_path() (dropbox.files.ListFolderContinueError method), 62
get_path() (dropbox.files.ListFolderError method), 63
get_path() (dropbox.files.ListRevisionsError method), 65
get_path() (dropbox.files.PreviewError method), 70
get_path() (dropbox.files.PropertiesError method), 71
get_path() (dropbox.files.SaveCopyReferenceError method), 78
get_path() (dropbox.files.SaveUrlError method), 80
get_path() (dropbox.files.SearchError method), 82
get_path() (dropbox.files.ThumbnailError method), 85

get_path() (dropbox.files.UploadError method), 88
get_path() (dropbox.files.UploadSessionFinishError method), 91
get_path() (dropbox.sharing.CreateSharedLinkError method), 106
get_path() (dropbox.sharing.CreateSharedLinkWithSettingsError method), 107
get_path() (dropbox.sharing.GetSharedLinksError method), 121
get_path() (dropbox.sharing.ListSharedLinksError method), 136
get_path_lookup() (dropbox.files.DeleteError method), 53
get_path_lookup() (dropbox.files.RestoreError method), 77
get_path_write() (dropbox.files.DeleteError method), 53
get_path_write() (dropbox.files.RestoreError method), 77
get_permission_denied_error() (dropbox.sharing.FileErrorResult method), 110
get_properties_error() (dropbox.files.AlphaGetMetadataError method), 49
get_properties_error() (dropbox.files.UploadErrorWithProperties method), 88
get_property_group_lookup() (dropbox.files.RemovePropertiesError method), 76
get_property_group_lookup() (dropbox.files.UpdatePropertiesError method), 87
get_relinquish_folder_membership_error() (dropbox.sharing.JobError method), 124
get_remove_folder_member_error() (dropbox.sharing.JobError method), 124
get_removed() (dropbox.team.TeamMemberStatus method), 241
get_result() (dropbox.sharing.ListFileMembersIndividualResult method), 131
get_settings_error() (dropbox.sharing.CreateSharedLinkWithSettingsError method), 107
get_settings_error() (dropbox.sharing.ModifySharedLinkSettingsError method), 142
get_success() (dropbox.files.DeleteBatchResultEntry method), 52
get_success() (dropbox.files.UploadSessionFinishBatchResultEntry method), 90
get_success() (dropbox.sharing.FileMemberActionIndividualResult method), 112
get_success() (dropbox.sharing.FileMemberRemoveActionResult method), 113
get_success() (dropbox.team.MemberAddResult method), 216
get_team() (dropbox.users.SpaceAllocation method), 250
get_team_license_limit() (dropbox.team.MemberAddResult method), 216
get_team_member_id() (dropbox.team.UserSelectorArg method), 243
get_team_member_ids() (dropbox.team.UsersSelectorArg method), 244
get_to() (dropbox.files.RelocationError method), 75
get_too_many_members() (dropbox.sharing.AddFolderMemberError method), 102
get_too_many_pending_invites() (dropbox.sharing.AddFolderMemberError method), 102
get_unshare_folder_error() (dropbox.sharing.JobError method), 125
get_unverified_dropbox_id() (dropbox.sharing.AddMemberSelectorError method), 104
get_update() (dropbox.files.WriteMode method), 96
get_user_already_on_team() (dropbox.team.MemberAddResult method), 216
get_user_already_paired() (dropbox.team.MemberAddResult method), 216
get_user_cannot_be_manager_of_company_managed_group() (dropbox.team.GroupMembersAddError method), 196
get_user_creation_failed() (dropbox.team.MemberAddResult method), 216
get_user_error() (dropbox.sharing.AddFileMemberError method), 99
get_user_error() (dropbox.sharing.GetFileMetadataError method), 119
get_user_error() (dropbox.sharing.ListFileMembersContinueError method), 129
get_user_error() (dropbox.sharing.ListFileMembersError method), 130
get_user_error() (dropbox.sharing.ListFilesContinueError method), 132
get_user_error() (dropbox.sharing.RemoveFileMemberError method), 150
get_user_error() (dropbox.sharing.UnshareFileError method), 175
get_user_migration_failed() (dropbox.team.MemberAddResult method), 216
get_user_on_another_team() (dropbox.team.MemberAddResult method), 216
get_users_not_found() (dropbox.team.GroupMembersAddError method), 196
get_web_session() (dropbox.team.RevokeDeviceSessionArg method),

235
GetAccountArg (class in dropbox.users), 248
GetAccountBatchArg (class in dropbox.users), 248
GetAccountBatchError (class in dropbox.users), 248
GetAccountError (class in dropbox.users), 249
GetActivityReport (class in dropbox.team), 188
GetCopyReferenceArg (class in dropbox.files), 58
GetCopyReferenceError (class in dropbox.files), 58
GetCopyReferenceResult (class in dropbox.files), 58
GetDevicesReport (class in dropbox.team), 190
GetFileMetadataArg (class in dropbox.sharing), 117
GetFileMetadataBatchArg (class in dropbox.sharing), 118
GetFileMetadataBatchResult (class in dropbox.sharing), 118
GetFileMetadataError (class in dropbox.sharing), 118
GetFileMetadataIndividualResult (class in dropbox.sharing), 119
GetMembershipReport (class in dropbox.team), 190
GetMetadataArg (class in dropbox.files), 59
GetMetadataArgs (class in dropbox.sharing), 120
GetMetadataError (class in dropbox.files), 59
GetSharedLinkFileArg (in module dropbox.sharing), 120
GetSharedLinkFileError (class in dropbox.sharing), 120
GetSharedLinkMetadataArg (class in dropbox.sharing), 121
GetSharedLinksArg (class in dropbox.sharing), 121
GetSharedLinksError (class in dropbox.sharing), 121
GetSharedLinksResult (class in dropbox.sharing), 122
GetStorageReport (class in dropbox.team), 191
GetTemporaryLinkArg (class in dropbox.files), 60
GetTemporaryLinkError (class in dropbox.files), 60
GetTemporaryLinkResult (class in dropbox.files), 60
given_name (dropbox.users.Name attribute), 250
GpsCoordinates (class in dropbox.files), 61
group (dropbox.sharing.GroupMembershipInfo attribute), 123
group (dropbox.team.GroupMembersAddArg attribute), 195
group (dropbox.team.GroupMemberSelector attribute), 195
group (dropbox.team.GroupMembersRemoveArg attribute), 198
group (dropbox.team.GroupMembersSelector attribute), 198
group (dropbox.team.GroupsMembersListArg attribute), 204
group (dropbox.team.GroupUpdateArgs attribute), 201
group_access (dropbox.sharing.RelinquishFileMembershipError attribute), 147
group_access (dropbox.sharing.RelinquishFolderMembershipError attribute), 148
group_access (dropbox.sharing.RemoveFolderMemberError attribute), 151
group_already_deleted (dropbox.team.GroupDeleteError attribute), 194
group_deleted (dropbox.sharing.AddMemberSelectorError attribute), 104
group_external_id (dropbox.team.GroupCreateArg attribute), 193
group_external_id() (dropbox.team.GroupSelector class method), 200
group_external_ids() (dropbox.team.GroupsSelector class method), 205
group_id() (dropbox.team.GroupSelector class method), 200
group_ids() (dropbox.team.GroupsSelector class method), 205
group_info (dropbox.team.GroupMembersChangeResult attribute), 197
group_info() (dropbox.team.GroupsGetInfoItem class method), 202
group_management_type (dropbox.team.GroupCreateArg attribute), 193
group_name (dropbox.team.GroupCreateArg attribute), 193
group_name_already_used (dropbox.team.GroupCreateError attribute), 193
group_name_invalid (dropbox.team.GroupCreateError attribute), 193
group_not_found (dropbox.team.GroupSelectorError attribute), 200
group_not_in_team (dropbox.team.GroupMembersAddError attribute), 196
group_not_in_team (dropbox.team.GroupMembersRemoveError attribute), 198
group_not_on_team (dropbox.sharing.AddMemberSelectorError attribute), 104
group_not_on_team (dropbox.team.GroupsGetInfoError attribute), 201
group_type (dropbox.sharing.GroupInfo attribute), 122
GroupAccessType (class in dropbox.team), 192
GroupCreateArg (class in dropbox.team), 192
GroupCreateError (class in dropbox.team), 193
GroupDeleteError (class in dropbox.team), 193
GroupFullInfo (class in dropbox.team), 194
GroupInfo (class in dropbox.sharing), 122
GroupMemberInfo (class in dropbox.team), 194
GroupMembersAddArg (class in dropbox.team), 195
GroupMembersAddError (class in dropbox.team), 196
GroupMembersChangeResult (class in dropbox.team), 197
GroupMemberSelector (class in dropbox.team), 194
GroupMemberSelectorError (class in dropbox.team), 195

GroupMemberSetAccessTypeError (class in dropbox.team), 195
GroupMembershipInfo (class in dropbox.sharing), 122
GroupMembersRemoveArg (class in dropbox.team), 198
GroupMembersRemoveError (class in dropbox.team), 198
GroupMembersSelector (class in dropbox.team), 198
GroupMembersSelectorError (class in dropbox.team), 198
GroupMembersSetAccessTypeArg (class in dropbox.team), 199
groups (dropbox.sharing.SharedFileMembers attribute), 160
groups (dropbox.sharing.SharedFolderMembers attribute), 164
groups (dropbox.team.GroupsListResult attribute), 203
groups (dropbox.team.TeamMemberProfile attribute), 240
GroupSelector (class in dropbox.team), 199
GroupSelectorError (class in dropbox.team), 200
GroupsGetInfoError (class in dropbox.team), 201
GroupsGetInfoItem (class in dropbox.team), 201
GroupsListArg (class in dropbox.team), 202
GroupsListContinueArg (class in dropbox.team), 202
GroupsListContinueError (class in dropbox.team), 203
GroupsListResult (class in dropbox.team), 203
GroupsMembersListArg (class in dropbox.team), 203
GroupsMembersListContinueArg (class in dropbox.team), 204
GroupsMembersListContinueError (class in dropbox.team), 204
GroupsMembersListResult (class in dropbox.team), 204
GroupsPollError (class in dropbox.team), 205
GroupsSelector (class in dropbox.team), 205
GroupUpdateArgs (class in dropbox.team), 200
GroupUpdateError (class in dropbox.team), 201

H

has_explicit_shared_members (dropbox.files.FileMetadata attribute), 55
has_more (dropbox.files.ListFolderResult attribute), 65
has_more (dropbox.sharing.ListSharedLinksResult attribute), 137
has_more (dropbox.team.GroupsListResult attribute), 203
has_more (dropbox.team.GroupsMembersListResult attribute), 205
has_more (dropbox.team.ListMembersAppsResult attribute), 209
has_more (dropbox.team.ListMembersDevicesResult attribute), 211
has_more (dropbox.team.ListTeamAppsResult attribute), 212
has_more (dropbox.team.ListTeamDevicesResult attribute), 213
has_more (dropbox.team.MembersListResult attribute), 225
height (dropbox.files.Dimensions attribute), 54
host_name (dropbox.team.DesktopClientSession attribute), 185
HttpError, 253

I

id (dropbox.files.FileMetadata attribute), 56
id (dropbox.files.FolderMetadata attribute), 57
id (dropbox.sharing.SharedFileMetadata attribute), 161
id (dropbox.sharing.SharedLinkMetadata attribute), 168
id (dropbox.users.Team attribute), 252
id_not_found() (dropbox.team.GroupsGetInfoItem class method), 202
id_not_found() (dropbox.team.MembersGetInfoItem class method), 223
in_progress (dropbox.async.PollResultBase attribute), 48
in_progress (dropbox.files.PreviewError attribute), 70
include_deleted (dropbox.files.GetMetadataArg attribute), 59
include_deleted (dropbox.files.ListFolderArg attribute), 62
include_desktop_clients (dropbox.team.ListMemberDevicesArg attribute), 207
include_desktop_clients (dropbox.team.ListMembersDevicesArg attribute), 210
include_desktop_clients (dropbox.team.ListTeamDevicesArg attribute), 212
include_has_explicit_shared_members (dropbox.files.GetMetadataArg attribute), 59
include_has_explicit_shared_members (dropbox.files.ListFolderArg attribute), 62
include_inherited (dropbox.sharing.ListFileMembersArg attribute), 127
include_media_info (dropbox.files.GetMetadataArg attribute), 59
include_media_info (dropbox.files.ListFolderArg attribute), 62
include_mobile_clients (dropbox.team.ListMemberDevicesArg attribute), 207
include_mobile_clients (dropbox.team.ListMembersDevicesArg attribute), 210
include_mobile_clients (dropbox.team.ListTeamDevicesArg attribute), 213

include_property_templates (dropbox.files.AlphaGetMetadataArg attribute), 49
include_removed (dropbox.team.MembersListArg attribute), 224
include_web_sessions (dropbox.team.ListMemberDevicesArg attribute), 207
include_web_sessions (dropbox.team.ListMembersDevicesArg attribute), 210
include_web_sessions (dropbox.team.ListTeamDevicesArg attribute), 213
IncludeMembersArg (class in dropbox.team), 206
incorrect_offset() (dropbox.files.UploadSessionLookupError class method), 92
individual() (dropbox.users.SpaceAllocation class method), 251
IndividualSpaceAllocation (class in dropbox.users), 249
initials (dropbox.sharing.MembershipInfo attribute), 141
inside_app_folder (dropbox.sharing.SharePathError attribute), 158
inside_osx_package (dropbox.sharing.SharePathError attribute), 158
inside_osx_package (dropbox.sharing.SharingFileAccessError attribute), 170
inside_public_folder (dropbox.sharing.SharePathError attribute), 158
inside_public_folder (dropbox.sharing.SharingFileAccessError attribute), 170
inside_shared_folder (dropbox.sharing.MountFolderError attribute), 143
inside_shared_folder (dropbox.sharing.SharePathError attribute), 158
insufficient_plan (dropbox.sharing.AddFolderMemberError attribute), 102
insufficient_plan (dropbox.sharing.UpdateFolderMemberError attribute), 177
insufficient_quota() (dropbox.sharing.MountFolderError class method), 143
insufficient_space (dropbox.files.WriteError attribute), 95
InsufficientQuotaAmounts (class in dropbox.sharing), 123
internal_error (dropbox.async.PollError attribute), 48
InternalServerError, 253
invalid_async_job_id (dropbox.async.PollError attribute), 48
invalid_comment (dropbox.sharing.AddFileMemberError attribute), 99
invalid_copy_reference (dropbox.files.SaveCopyReferenceError attribute), 79
invalid_cursor (dropbox.sharing.ListFileMembersContinueError attribute), 129
invalid_cursor (dropbox.sharing.ListFilesContinueError attribute), 132
invalid_cursor (dropbox.sharing.ListFolderMembersContinueError attribute), 134
invalid_cursor (dropbox.sharing.ListFoldersContinueError attribute), 135
invalid_cursor (dropbox.team.GroupsListContinueError attribute), 203
invalid_cursor (dropbox.team.GroupsMembersListContinueError attribute), 204
invalid_cursor (dropbox.team.MembersListContinueError attribute), 224
invalid_dropbox_id (dropbox.sharing.SharedFolderMemberError attribute), 163
invalid_dropbox_id (dropbox.sharing.TransferFolderError attribute), 172
invalid_dropbox_id() (dropbox.sharing.AddMemberSelectorError class method), 104
invalid_email() (dropbox.sharing.AddMemberSelectorError class method), 104
invalid_file (dropbox.sharing.SharingFileAccessError attribute), 170
invalid_file_action_error() (dropbox.sharing.FileErrorResult class method), 110
invalid_id (dropbox.sharing.SharedFolderAccessError attribute), 162
invalid_member (dropbox.sharing.FileMemberActionError attribute), 111
invalid_path (dropbox.sharing.SharePathError attribute), 158
invalid_path_root() (dropbox.files.LookupError class method), 67
invalid_path_root() (dropbox.sharing.SharePathError class method), 158
invalid_revision (dropbox.files.RestoreError attribute), 77
invalid_settings (dropbox.sharing.SharedLinkSettingsError attribute), 170
invalid_url (dropbox.files.SaveUrlError attribute), 80
InvalidPropertyGroupError (class in dropbox.files), 61
invite_editor (dropbox.sharing.FolderAction attribute), 115
invite_viewer (dropbox.sharing.FileAction attribute), 108

invite_viewer (dropbox.sharing.FolderAction attribute), 115
invite_viewer_no_comment (dropbox.sharing.FileAction attribute), 108
invite_viewer_no_comment (dropbox.sharing.FolderAction attribute), 115
invited (dropbox.team.TeamMemberStatus attribute), 241
invitee (dropbox.sharing.InviteeMembershipInfo attribute), 124
InviteeInfo (class in dropbox.sharing), 123
InviteeMembershipInfo (class in dropbox.sharing), 124
invitees (dropbox.sharing.SharedFileMembers attribute), 160
invitees (dropbox.sharing.SharedFolderMembers attribute), 164
ios (dropbox.team.DevicesActive attribute), 187
ip_address (dropbox.team.DeviceSession attribute), 186
ipad (dropbox.team.MobileClientPlatform attribute), 233
iphone (dropbox.team.MobileClientPlatform attribute), 233
is_access_denied() (dropbox.sharing.CreateSharedLinkWithSettingsError method), 107
is_access_denied() (dropbox.team.GroupsPollError method), 205
is_access_error() (dropbox.sharing.AddFileMemberError method), 99
is_access_error() (dropbox.sharing.AddFolderMemberError method), 102
is_access_error() (dropbox.sharing.FileMemberActionError method), 111
is_access_error() (dropbox.sharing.GetFileMetadataError method), 119
is_access_error() (dropbox.sharing.GetFileMetadataIndividualResult method), 120
is_access_error() (dropbox.sharing.ListFileMembersContinueError method), 129
is_access_error() (dropbox.sharing.ListFileMembersError method), 130
is_access_error() (dropbox.sharing.ListFileMembersIndividualResult method), 131
is_access_error() (dropbox.sharing.ListFolderMembersContinueError method), 134
is_access_error() (dropbox.sharing.MountFolderError method), 143
is_access_error() (dropbox.sharing.RelinquishFileMembershipError method), 147
is_access_error() (dropbox.sharing.RelinquishFolderMembershipError method), 148
is_access_error() (dropbox.sharing.RemoveFileMemberError method), 150
is_access_error() (dropbox.sharing.RemoveFolderMemberError method), 151
is_access_error() (dropbox.sharing.TransferFolderError method), 172
is_access_error() (dropbox.sharing.UnmountFolderError method), 174
is_access_error() (dropbox.sharing.UnshareFileError method), 175
is_access_error() (dropbox.sharing.UnshareFolderError method), 176
is_access_error() (dropbox.sharing.UpdateFolderMemberError method), 177
is_access_error() (dropbox.sharing.UpdateFolderPolicyError method), 179
is_active() (dropbox.team.TeamMemberStatus method), 241
is_add() (dropbox.files.WriteMode method), 96
is_already_mounted() (dropbox.sharing.MountFolderError method), 143
is_already_shared() (dropbox.sharing.SharePathError method), 158
is_android() (dropbox.team.MobileClientPlatform method), 233
is_anyone() (dropbox.sharing.MemberPolicy method), 140
is_anyone() (dropbox.sharing.SharedLinkPolicy method), 169
is_app_folder (dropbox.sharing.SharePathError attribute), 158
is_app_folder (dropbox.team.ApiApp attribute), 183
is_app_not_found() (dropbox.team.RevokeLinkedAppError method), 238
is_async_job_id() (dropbox.async.LaunchResultBase method), 47
is_automatic_group() (dropbox.sharing.AddMemberSelectorError method), 104
is_bad_member() (dropbox.sharing.AddFolderMemberError method), 102
is_bad_path() (dropbox.sharing.ShareFolderErrorBase method), 156

is_basic() (dropbox.users.AccountType method), 246
is_blackberry() (dropbox.team.MobileClientPlatform method), 233
is_both() (dropbox.files.SearchMatchType method), 83
is_business() (dropbox.users.AccountType method), 246
is_cannot_keep_account_and_delete_data() (dropbox.team.MembersRemoveError method), 227
is_cannot_keep_account_and_transfer() (dropbox.team.MembersRemoveError method), 227
is_cannot_set_permissions() (dropbox.team.MembersSetPermissionsError method), 229
is_cant_copy_shared_folder() (dropbox.files.RelocationError method), 75
is_cant_move_folder_into_itself() (dropbox.files.RelocationError method), 75
is_cant_nest_shared_folder() (dropbox.files.RelocationError method), 75
is_cant_share_outside_team() (dropbox.sharing.AddFolderMemberError method), 102
is_change_options() (dropbox.sharing.FolderAction method), 115
is_closed() (dropbox.files.UploadSessionLookupError method), 92
is_complete() (dropbox.async.LaunchEmptyResult method), 46
is_complete() (dropbox.async.PollEmptyResult method), 47
is_complete() (dropbox.files.DeleteBatchJobStatus method), 52
is_complete() (dropbox.files.RelocationBatchJobStatus method), 74
is_complete() (dropbox.files.SaveUrlJobStatus method), 81
is_complete() (dropbox.files.SaveUrlResult method), 81
is_complete() (dropbox.files.UploadSessionFinishBatchJobStatus method), 90
is_complete() (dropbox.sharing.JobStatus method), 126
is_complete() (dropbox.sharing.RemoveMemberJobStatus method), 153
is_complete() (dropbox.sharing.ShareFolderJobStatus method), 157
is_complete() (dropbox.sharing.ShareFolderLaunch method), 157
is_complete() (dropbox.team.MembersAddJobStatus method), 221
is_complete() (dropbox.team.MembersAddLaunch method), 221
is_conflict() (dropbox.files.WriteError method), 95
is_contains_app_folder() (dropbox.sharing.SharePathError method), 158
is_contains_shared_folder() (dropbox.sharing.SharePathError method), 159
is_contains_team_folder() (dropbox.sharing.SharePathError method), 159
is_content() (dropbox.files.SearchMatchType method), 83
is_conversion_error() (dropbox.files.ThumbnailError method), 85
is_create_link() (dropbox.sharing.FileAction method), 108
is_create_link() (dropbox.sharing.FolderAction method), 115
is_delete_on_unlink_supported (dropbox.team.DesktopClientSession attribute), 185
is_deleted (dropbox.files.ListRevisionsResult attribute), 66
is_deleted_filename() (dropbox.files.SearchMode method), 84
is_desktop_client() (dropbox.team.RevokeDeviceSessionArg method), 235
is_device_session_not_found() (dropbox.team.RevokeDeviceSessionError method), 236
is_disallowed_name() (dropbox.files.WriteError method), 95
is_disallowed_shared_link_policy() (dropbox.sharing.ShareFolderErrorBase method), 156
is_disallowed_shared_link_policy() (dropbox.sharing.UpdateFolderPolicyError method), 179
is_does_not_fit_template() (dropbox.files.InvalidPropertyGroupError method), 61
is_download_failed() (dropbox.files.SaveUrlError method), 80
is_dropbox_id() (dropbox.sharing.MemberSelector method), 141
is_duplicate_external_member_id() (dropbox.team.MemberAddResult method), 216
is_duplicate_user() (dropbox.team.GroupMembersAddError method), 196
is_duplicated_or_nested_paths() (dropbox.files.RelocationBatchError method), 73
is_edit_contents() (dropbox.sharing.FileAction method), 108
is_edit_contents() (dropbox.sharing.FolderAction method), 115
is_editor() (dropbox.sharing.AccessLevel method), 97

is_editors() (dropbox.sharing.AclUpdatePolicy method), 98
is_email() (dropbox.sharing.InviteeInfo method), 123
is_email() (dropbox.sharing.MemberSelector method), 141
is_email() (dropbox.team.UserSelectorArg method), 243
is_email_address_too_long_to_be_disabled() (dropbox.team.MembersRemoveError method), 227
is_email_not_verified() (dropbox.sharing.CreateSharedLinkWithSettingsError method), 107
is_email_not_verified() (dropbox.sharing.ModifySharedLinkSettingsError method), 142
is_email_reserved_for_other_user() (dropbox.team.MembersSetProfileError method), 231
is_email_unverified() (dropbox.sharing.AddFolderMemberError method), 102
is_email_unverified() (dropbox.sharing.SharedFolderAccessError method), 162
is_email_unverified() (dropbox.sharing.ShareFolderErrorBase method), 156
is_email_unverified() (dropbox.sharing.SharingUserError method), 171
is_email_verify_required() (dropbox.sharing.SharedLinkAccessFailureReason method), 166
is_emails() (dropbox.team.UsersSelectorArg method), 244
is_external_id() (dropbox.team.UserSelectorArg method), 243
is_external_id_already_in_use() (dropbox.team.GroupCreateError method), 193
is_external_id_already_in_use() (dropbox.team.GroupUpdateError method), 201
is_external_id_and_new_external_id_unsafe() (dropbox.team.MembersSetProfileError method), 231
is_external_id_used_by_other_user() (dropbox.team.MembersSetProfileError method), 231
is_external_ids() (dropbox.team.UsersSelectorArg method), 245
is_failed() (dropbox.files.DeleteBatchJobStatus method), 52
is_failed() (dropbox.files.RelocationBatchJobStatus method), 74
is_failed() (dropbox.files.SaveUrlJobStatus method), 81
is_failed() (dropbox.sharing.JobStatus method), 126
is_failed() (dropbox.sharing.RemoveMemberJobStatus method), 153
is_failed() (dropbox.sharing.ShareFolderJobStatus method), 157
is_failed() (dropbox.team.MembersAddJobStatus method), 221
is_failure() (dropbox.files.DeleteBatchResultEntry method), 53
is_failure() (dropbox.files.UploadSessionFinishBatchResultEntry method), 91
is_file (dropbox.sharing.SharePathError attribute), 159
is_file() (dropbox.files.WriteConflictError method), 94
is_file() (dropbox.sharing.PendingUploadMode method), 145
is_file_ancestor() (dropbox.files.WriteConflictError method), 94
is_file_not_found_error() (dropbox.sharing.FileErrorResult method), 110
is_filename() (dropbox.files.SearchMatchType method), 83
is_filename() (dropbox.files.SearchMode method), 84
is_filename_and_content() (dropbox.files.SearchMode method), 84
is_folder (dropbox.sharing.SharingFileAccessError attribute), 170
is_folder() (dropbox.files.WriteConflictError method), 94
is_folder() (dropbox.sharing.PendingUploadMode method), 145
is_folder_is_limited_team_folder() (dropbox.sharing.PermissionDeniedReason method), 146
is_folder_owner() (dropbox.sharing.RelinquishFolderMembershipError method), 148
is_folder_owner() (dropbox.sharing.RemoveFolderMemberError method), 151
is_free_team_member_limit_reached() (dropbox.team.MemberAddResult method), 217
is_from_lookup() (dropbox.files.RelocationError method), 75
is_from_write() (dropbox.files.RelocationError method), 75
is_full() (dropbox.team.TeamMembershipType method), 242
is_group_access() (dropbox.sharing.RelinquishFileMembershipError method), 147
is_group_access() (dropbox.sharing.RelinquishFolderMembershipError method), 148
is_group_access() (dropbox.sharing.RemoveFolderMemberError method), 151

is_group_already_deleted() (dropbox.team.GroupDeleteError method), 194
is_group_deleted() (dropbox.sharing.AddMemberSelectorError method), 104
is_group_external_id() (dropbox.team.GroupSelector method), 200
is_group_external_ids() (dropbox.team.GroupsSelector method), 206
is_group_id() (dropbox.team.GroupSelector method), 200
is_group_ids() (dropbox.team.GroupsSelector method), 206
is_group_info() (dropbox.team.GroupsGetInfoItem method), 202
is_group_name_already_used() (dropbox.team.GroupCreateError method), 193
is_group_name_invalid() (dropbox.team.GroupCreateError method), 193
is_group_not_found() (dropbox.team.GroupSelectorError method), 200
is_group_not_in_team() (dropbox.team.GroupMembersAddError method), 196
is_group_not_in_team() (dropbox.team.GroupMembersRemoveError method), 198
is_group_not_on_team() (dropbox.sharing.AddMemberSelectorError method), 104
is_group_not_on_team() (dropbox.team.GroupsGetInfoError method), 201
is_id_not_found() (dropbox.team.GroupsGetInfoItem method), 202
is_id_not_found() (dropbox.team.MembersGetInfoItem method), 223
is_in_progress() (dropbox.async.PollResultBase method), 48
is_in_progress() (dropbox.files.PreviewError method), 70
is_incorrect_offset() (dropbox.files.UploadSessionLookupError method), 92
is_individual() (dropbox.users.SpaceAllocation method), 251
is_inherited (dropbox.sharing.MembershipInfo attribute), 141
is_inside_app_folder() (dropbox.sharing.SharePathError method), 159
is_inside_osx_package() (dropbox.sharing.SharePathError method), 159
is_inside_osx_package() (dropbox.sharing.SharingFileAccessError method), 170
is_inside_public_folder() (dropbox.sharing.SharePathError method), 159
is_inside_public_folder() (dropbox.sharing.SharingFileAccessError method), 170
is_inside_shared_folder() (dropbox.sharing.MountFolderError method), 144
is_inside_shared_folder() (dropbox.sharing.SharePathError method), 159
is_insufficient_plan() (dropbox.sharing.AddFolderMemberError method), 102
is_insufficient_plan() (dropbox.sharing.UpdateFolderMemberError method), 177
is_insufficient_quota() (dropbox.sharing.MountFolderError method), 144
is_insufficient_space() (dropbox.files.WriteError method), 95
is_internal_error() (dropbox.async.PollError method), 48
is_invalid_async_job_id() (dropbox.async.PollError method), 48
is_invalid_comment() (dropbox.sharing.AddFileMemberError method), 100
is_invalid_copy_reference() (dropbox.files.SaveCopyReferenceError method), 79
is_invalid_cursor() (dropbox.sharing.ListFileMembersContinueError method), 129
is_invalid_cursor() (dropbox.sharing.ListFilesContinueError method), 132
is_invalid_cursor() (dropbox.sharing.ListFolderMembersContinueError method), 134
is_invalid_cursor() (dropbox.sharing.ListFoldersContinueError method), 135
is_invalid_cursor() (dropbox.team.GroupsListContinueError method), 203
is_invalid_cursor() (dropbox.team.GroupsMembersListContinueError method), 204
is_invalid_cursor() (dropbox.team.MembersListContinueError method), 224
is_invalid_dropbox_id() (dropbox.sharing.AddMemberSelectorError method), 105

is_invalid_dropbox_id() (dropbox.sharing.SharedFolderMemberError method), 163
is_invalid_dropbox_id() (dropbox.sharing.TransferFolderError method), 172
is_invalid_email() (dropbox.sharing.AddMemberSelectorError method), 105
is_invalid_file() (dropbox.sharing.SharingFileAccessError method), 170
is_invalid_file_action_error() (dropbox.sharing.FileErrorResult method), 110
is_invalid_id() (dropbox.sharing.SharedFolderAccessError method), 162
is_invalid_member() (dropbox.sharing.FileMemberActionError method), 112
is_invalid_path() (dropbox.sharing.SharePathError method), 159
is_invalid_path_root() (dropbox.files.LookupError method), 67
is_invalid_path_root() (dropbox.sharing.SharePathError method), 159
is_invalid_revision() (dropbox.files.RestoreError method), 77
is_invalid_settings() (dropbox.sharing.SharedLinkSettingsError method), 170
is_invalid_url() (dropbox.files.SaveUrlError method), 80
is_invite_editor() (dropbox.sharing.FolderAction method), 115
is_invite_viewer() (dropbox.sharing.FileAction method), 109
is_invite_viewer() (dropbox.sharing.FolderAction method), 115
is_invite_viewer_no_comment() (dropbox.sharing.FileAction method), 109
is_invite_viewer_no_comment() (dropbox.sharing.FolderAction method), 115
is_invited() (dropbox.team.TeamMemberStatus method), 241
is_ipad() (dropbox.team.MobileClientPlatform method), 233
is_iphone() (dropbox.team.MobileClientPlatform method), 233
is_is_app_folder() (dropbox.sharing.SharePathError method), 159
is_is_file() (dropbox.sharing.SharePathError method), 159
is_is_folder() (dropbox.sharing.SharingFileAccessError method), 170
is_is_osx_package() (dropbox.sharing.SharePathError method), 159
is_is_public_folder() (dropbox.sharing.SharePathError method), 159
is_jpeg() (dropbox.files.ThumbnailFormat method), 86
is_last_admin() (dropbox.team.MembersSetPermissionsError method), 229
is_leave_a_copy() (dropbox.sharing.FolderAction method), 115
is_leave_a_copy() (dropbox.sharing.MemberAction method), 138
is_limited() (dropbox.team.TeamMembershipType method), 242
is_linux() (dropbox.team.DesktopPlatform method), 186
is_login_required() (dropbox.sharing.SharedLinkAccessFailureReason method), 166
is_lookup_failed() (dropbox.files.UploadSessionFinishError method), 91
is_mac() (dropbox.team.DesktopPlatform method), 186
is_make_editor() (dropbox.sharing.MemberAction method), 138
is_make_owner() (dropbox.sharing.MemberAction method), 138
is_make_viewer() (dropbox.sharing.MemberAction method), 139
is_make_viewer_no_comment() (dropbox.sharing.MemberAction method), 139
is_malformed_path() (dropbox.files.LookupError method), 67
is_malformed_path() (dropbox.files.WriteError method), 95
is_member() (dropbox.team.GroupAccessType method), 192
is_member_error() (dropbox.sharing.FileMemberActionIndividualResult method), 112
is_member_error() (dropbox.sharing.FileMemberRemoveActionResult method), 113
is_member_error() (dropbox.sharing.RemoveFolderMemberError method), 152
is_member_error() (dropbox.sharing.UpdateFolderMemberError method), 177
is_member_info() (dropbox.team.MembersGetInfoItem method), 223
is_member_not_found() (dropbox.team.ListMemberAppsError method), 206
is_member_not_found() (dropbox.team.ListMemberDevicesError method), 207

is_member_not_found() (dropbox.team.RevokeDeviceSessionError method), 237
is_member_not_found() (dropbox.team.RevokeLinkedAppError method), 238
is_member_not_in_group() (dropbox.team.GroupMemberSelectorError method), 195
is_member_not_in_group() (dropbox.team.GroupMembersSelectorError method), 199
is_member_only() (dropbox.team.AdminTier method), 183
is_members() (dropbox.sharing.SharedLinkPolicy method), 169
is_members_not_in_team() (dropbox.team.GroupMembersAddError method), 197
is_metadata() (dropbox.files.MediaInfo method), 68
is_metadata() (dropbox.sharing.GetFileMetadataIndividualResult method), 120
is_mobile_client() (dropbox.team.RevokeDeviceSessionArg method), 236
is_mounted() (dropbox.sharing.RelinquishFolderMembershipError method), 148
is_new_owner_email_unverified() (dropbox.sharing.TransferFolderError method), 173
is_new_owner_not_a_member() (dropbox.sharing.TransferFolderError method), 173
is_new_owner_unmounted() (dropbox.sharing.TransferFolderError method), 173
is_no_account() (dropbox.users.GetAccountBatchError method), 249
is_no_account() (dropbox.users.GetAccountError method), 249
is_no_explicit_access() (dropbox.sharing.RemoveFileMemberError method), 150
is_no_explicit_access() (dropbox.sharing.SharedFolderMemberError method), 163
is_no_explicit_access() (dropbox.sharing.UpdateFolderMemberError method), 177
is_no_new_data_specified() (dropbox.team.MembersSetProfileError method), 231
is_no_permission() (dropbox.files.SaveCopyReferenceError method), 79
is_no_permission() (dropbox.sharing.AddFolderMemberError method), 102
is_no_permission() (dropbox.sharing.FileMemberActionError method), 112
is_no_permission() (dropbox.sharing.MountFolderError method), 144
is_no_permission() (dropbox.sharing.RelinquishFileMembershipError method), 147
is_no_permission() (dropbox.sharing.RelinquishFolderMembershipError method), 148
is_no_permission() (dropbox.sharing.RemoveFolderMemberError method), 152
is_no_permission() (dropbox.sharing.ShareFolderError method), 155
is_no_permission() (dropbox.sharing.SharingFileAccessError method), 171
is_no_permission() (dropbox.sharing.TransferFolderError method), 173
is_no_permission() (dropbox.sharing.UnmountFolderError method), 174
is_no_permission() (dropbox.sharing.UnshareFolderError method), 176
is_no_permission() (dropbox.sharing.UpdateFolderMemberError method), 178
is_no_permission() (dropbox.sharing.UpdateFolderPolicyError method), 179
is_no_write_permission() (dropbox.files.WriteError method), 95
is_not_a_member() (dropbox.sharing.SharedFolderAccessError method), 162
is_not_a_member() (dropbox.sharing.SharedFolderMemberError method), 163
is_not_authorized() (dropbox.sharing.SharedLinkSettingsError method), 170
is_not_closed() (dropbox.files.UploadSessionLookupError method), 92
is_not_file() (dropbox.files.LookupError method), 67
is_not_folder() (dropbox.files.LookupError method), 67
is_not_found() (dropbox.files.LookupError method), 67

is_not_found() (dropbox.files.SaveCopyReferenceError method), 79
is_not_found() (dropbox.files.SaveUrlError method), 80
is_not_found() (dropbox.files.UploadSessionLookupError method), 93
is_not_mountable() (dropbox.sharing.MountFolderError method), 144
is_not_on_team() (dropbox.sharing.UpdateFolderPolicyError method), 179
is_not_unmountable() (dropbox.sharing.UnmountFolderError method), 174
is_osx_package (dropbox.sharing.SharePathError attribute), 159
is_other() (dropbox.async.PollError method), 48
is_other() (dropbox.files.DeleteBatchError method), 51
is_other() (dropbox.files.DeleteBatchJobStatus method), 52
is_other() (dropbox.files.DeleteError method), 53
is_other() (dropbox.files.DownloadError method), 54
is_other() (dropbox.files.GetCopyReferenceError method), 58
is_other() (dropbox.files.GetTemporaryLinkError method), 60
is_other() (dropbox.files.ListFolderContinueError method), 62
is_other() (dropbox.files.ListFolderError method), 63
is_other() (dropbox.files.ListFolderLongpollError method), 64
is_other() (dropbox.files.ListRevisionsError method), 65
is_other() (dropbox.files.LookupError method), 67
is_other() (dropbox.files.RelocationError method), 75
is_other() (dropbox.files.RestoreError method), 77
is_other() (dropbox.files.SaveCopyReferenceError method), 79
is_other() (dropbox.files.SaveUrlError method), 80
is_other() (dropbox.files.SearchError method), 82
is_other() (dropbox.files.UploadError method), 88
is_other() (dropbox.files.UploadSessionFinishError method), 91
is_other() (dropbox.files.UploadSessionLookupError method), 93
is_other() (dropbox.files.WriteConflictError method), 94
is_other() (dropbox.files.WriteError method), 95
is_other() (dropbox.sharing.AccessLevel method), 97
is_other() (dropbox.sharing.AclUpdatePolicy method), 98
is_other() (dropbox.sharing.AddFileMemberError method), 100
is_other() (dropbox.sharing.AddFolderMemberError method), 102
is_other() (dropbox.sharing.AddMemberSelectorError method), 105
is_other() (dropbox.sharing.CreateSharedLinkError method), 106
is_other() (dropbox.sharing.FileAction method), 109
is_other() (dropbox.sharing.FileErrorResult method), 110
is_other() (dropbox.sharing.FileMemberActionError method), 112
is_other() (dropbox.sharing.FileMemberRemoveActionResult method), 113
is_other() (dropbox.sharing.FolderAction method), 115
is_other() (dropbox.sharing.GetFileMetadataError method), 119
is_other() (dropbox.sharing.GetFileMetadataIndividualResult method), 120
is_other() (dropbox.sharing.GetSharedLinksError method), 121
is_other() (dropbox.sharing.InviteeInfo method), 123
is_other() (dropbox.sharing.JobError method), 125
is_other() (dropbox.sharing.ListFileMembersContinueError method), 129
is_other() (dropbox.sharing.ListFileMembersError method), 130
is_other() (dropbox.sharing.ListFileMembersIndividualResult method), 131
is_other() (dropbox.sharing.ListFilesContinueError method), 132
is_other() (dropbox.sharing.ListFolderMembersContinueError method), 134
is_other() (dropbox.sharing.ListFoldersContinueError method), 135
is_other() (dropbox.sharing.ListSharedLinksError method), 137
is_other() (dropbox.sharing.MemberAction method), 139
is_other() (dropbox.sharing.MemberPolicy method), 140
is_other() (dropbox.sharing.MemberSelector method), 141
is_other() (dropbox.sharing.MountFolderError method), 144
is_other() (dropbox.sharing.PermissionDeniedReason method), 146
is_other() (dropbox.sharing.RelinquishFileMembershipError method), 147
is_other() (dropbox.sharing.RelinquishFolderMembershipError method), 148
is_other() (dropbox.sharing.RemoveFileMemberError method), 150
is_other() (dropbox.sharing.RemoveFolderMemberError method), 152
is_other() (dropbox.sharing.ResolvedVisibility method), 154
is_other() (dropbox.sharing.SharedFolderAccessError method), 162
is_other() (dropbox.sharing.SharedFolderMemberError method), 163

is_other() (dropbox.sharing.SharedLinkAccessFailureReason method), 166
is_other() (dropbox.sharing.SharedLinkError method), 167
is_other() (dropbox.sharing.SharedLinkPolicy method), 169
is_other() (dropbox.sharing.ShareFolderErrorBase method), 156
is_other() (dropbox.sharing.SharePathError method), 160
is_other() (dropbox.sharing.SharingFileAccessError method), 171
is_other() (dropbox.sharing.SharingUserError method), 171
is_other() (dropbox.sharing.TransferFolderError method), 173
is_other() (dropbox.sharing.UnmountFolderError method), 174
is_other() (dropbox.sharing.UnshareFileError method), 175
is_other() (dropbox.sharing.UnshareFolderError method), 176
is_other() (dropbox.sharing.UpdateFolderMemberError method), 178
is_other() (dropbox.sharing.UpdateFolderPolicyError method), 179
is_other() (dropbox.sharing.Visibility method), 181
is_other() (dropbox.team.DateRangeError method), 184
is_other() (dropbox.team.DesktopPlatform method), 186
is_other() (dropbox.team.GroupCreateError method), 193
is_other() (dropbox.team.GroupSelectorError method), 200
is_other() (dropbox.team.GroupsGetInfoError method), 201
is_other() (dropbox.team.GroupsListContinueError method), 203
is_other() (dropbox.team.GroupsMembersListContinueError method), 204
is_other() (dropbox.team.ListMemberAppsError method), 206
is_other() (dropbox.team.ListMemberDevicesError method), 207
is_other() (dropbox.team.ListMembersAppsError method), 209
is_other() (dropbox.team.ListMembersDevicesError method), 210
is_other() (dropbox.team.ListTeamAppsError method), 211
is_other() (dropbox.team.ListTeamDevicesError method), 213
is_other() (dropbox.team.MembersDeactivateError method), 222
is_other() (dropbox.team.MembersGetInfoError method), 222
is_other() (dropbox.team.MembersListContinueError method), 224
is_other() (dropbox.team.MembersListError method), 224
is_other() (dropbox.team.MembersRecoverError method), 225
is_other() (dropbox.team.MembersSendWelcomeError method), 228
is_other() (dropbox.team.MembersSetPermissionsError method), 229
is_other() (dropbox.team.MembersSetProfileError method), 231
is_other() (dropbox.team.MobileClientPlatform method), 233
is_other() (dropbox.team.RevokeDeviceSessionBatchError method), 236
is_other() (dropbox.team.RevokeDeviceSessionError method), 237
is_other() (dropbox.team.RevokeLinkedAppBatchError method), 238
is_other() (dropbox.team.RevokeLinkedAppError method), 238
is_other() (dropbox.users.GetAccountBatchError method), 249
is_other() (dropbox.users.GetAccountError method), 249
is_other() (dropbox.users.SpaceAllocation method), 251
is_overwrite() (dropbox.files.WriteMode method), 96
is_owner (dropbox.sharing.GroupInfo attribute), 122
is_owner() (dropbox.sharing.AccessLevel method), 97
is_owner() (dropbox.sharing.AclUpdatePolicy method), 98
is_owner() (dropbox.team.GroupAccessType method), 192
is_owner_only() (dropbox.sharing.SharedLinkAccessFailureReason method), 166
is_paired (dropbox.users.FullAccount attribute), 247
is_param_cannot_be_empty() (dropbox.team.MembersSetProfileError method), 231
is_password() (dropbox.sharing.RequestedVisibility method), 153
is_password() (dropbox.sharing.Visibility method), 181
is_password_required() (dropbox.sharing.SharedLinkAccessFailureReason method), 166
is_path() (dropbox.files.CreateFolderError method), 50
is_path() (dropbox.files.DownloadError method), 55
is_path() (dropbox.files.GetCopyReferenceError method), 58
is_path() (dropbox.files.GetMetadataError method), 60
is_path() (dropbox.files.GetTemporaryLinkError method), 60

is_path() (dropbox.files.ListFolderContinueError method), 62
is_path() (dropbox.files.ListFolderError method), 63
is_path() (dropbox.files.ListRevisionsError method), 66
is_path() (dropbox.files.PreviewError method), 70
is_path() (dropbox.files.PropertiesError method), 71
is_path() (dropbox.files.SaveCopyReferenceError method), 79
is_path() (dropbox.files.SaveUrlError method), 80
is_path() (dropbox.files.SearchError method), 82
is_path() (dropbox.files.ThumbnailError method), 86
is_path() (dropbox.files.UploadError method), 88
is_path() (dropbox.files.UploadSessionFinishError method), 91
is_path() (dropbox.sharing.CreateSharedLinkError method), 106
is_path() (dropbox.sharing.CreateSharedLinkWithSettingsError method), 107
is_path() (dropbox.sharing.GetSharedLinksError method), 121
is_path() (dropbox.sharing.ListSharedLinksError method), 137
is_path_lookup() (dropbox.files.DeleteError method), 53
is_path_lookup() (dropbox.files.RestoreError method), 77
is_path_write() (dropbox.files.DeleteError method), 53
is_path_write() (dropbox.files.RestoreError method), 78
is_pending() (dropbox.files.MediaInfo method), 68
is_permission_denied_error() (dropbox.sharing.FileErrorResult method), 110
is_png() (dropbox.files.ThumbnailFormat method), 86
is_pro() (dropbox.users.AccountType method), 246
is_properties_error() (dropbox.files.AlphaGetMetadataError method), 49
is_properties_error() (dropbox.files.UploadErrorWithProperties method), 88
is_property_field_too_large() (dropbox.files.InvalidPropertyGroupError method), 61
is_property_group_already_exists() (dropbox.files.AddPropertiesError method), 48
is_property_group_lookup() (dropbox.files.RemovePropertiesError method), 76
is_property_group_lookup() (dropbox.files.UpdatePropertiesError method), 87
is_property_group_not_found() (dropbox.files.LookUpPropertiesError method), 66
is_public() (dropbox.sharing.RequestedVisibility method), 153
is_public() (dropbox.sharing.Visibility method), 181
is_public_folder (dropbox.sharing.SharePathError attribute), 160
is_rate_limit() (dropbox.sharing.AddFileMemberError method), 100
is_rate_limit() (dropbox.sharing.AddFolderMemberError method), 102
is_recoverable (dropbox.team.RemovedStatus attribute), 235
is_relinquish_folder_membership_error() (dropbox.sharing.JobError method), 125
is_relinquish_membership() (dropbox.sharing.FileAction method), 109
is_relinquish_membership() (dropbox.sharing.FolderAction method), 116
is_remove() (dropbox.sharing.MemberAction method), 139
is_remove_folder_member_error() (dropbox.sharing.JobError method), 125
is_remove_last_admin() (dropbox.team.MembersRemoveError method), 227
is_removed() (dropbox.team.TeamMemberStatus method), 241
is_removed_and_transfer_admin_should_differ() (dropbox.team.MembersRemoveError method), 227
is_removed_and_transfer_dest_should_differ() (dropbox.team.MembersRemoveError method), 227
is_reset() (dropbox.files.ListFolderContinueError method), 63
is_reset() (dropbox.files.ListFolderLongpollError method), 64
is_reset() (dropbox.sharing.ListSharedLinksError method), 137
is_reset() (dropbox.team.ListMembersAppsError method), 209
is_reset() (dropbox.team.ListMembersDevicesError method), 210
is_reset() (dropbox.team.ListTeamAppsError method), 211
is_reset() (dropbox.team.ListTeamDevicesError method), 213
is_restricted_content() (dropbox.files.LookupError method), 67
is_result() (dropbox.sharing.ListFileMembersIndividualResult method), 131
is_set_profile_disallowed() (dropbox.team.MembersSetProfileError method), 231
is_settings_error() (dropbox.sharing.CreateSharedLinkWithSettingsError method), 108

is_settings_error() (dropbox.sharing.ModifySharedLinkSettingsError method), 142
is_share_link() (dropbox.sharing.FileAction method), 109
is_share_link() (dropbox.sharing.FolderAction method), 116
is_shared_folder_only() (dropbox.sharing.ResolvedVisibility method), 154
is_shared_folder_only() (dropbox.sharing.Visibility method), 181
is_shared_link_access_denied() (dropbox.sharing.SharedLinkError method), 167
is_shared_link_already_exists() (dropbox.sharing.CreateSharedLinkWithSettingsError method), 108
is_shared_link_is_directory() (dropbox.sharing.GetSharedLinkFileError method), 120
is_shared_link_malformed() (dropbox.sharing.RevokeSharedLinkError method), 154
is_shared_link_not_found() (dropbox.sharing.SharedLinkError method), 167
is_success() (dropbox.files.DeleteBatchResultEntry method), 53
is_success() (dropbox.files.UploadSessionFinishBatchResultEntry method), 91
is_success() (dropbox.sharing.FileMemberActionIndividualResult method), 112
is_success() (dropbox.sharing.FileMemberRemoveActionResult method), 114
is_success() (dropbox.team.MemberAddResult method), 217
is_support_admin() (dropbox.team.AdminTier method), 183
is_suspend_inactive_user() (dropbox.team.MembersSuspendError method), 232
is_suspend_last_admin() (dropbox.team.MembersSuspendError method), 232
is_suspended() (dropbox.team.TeamMemberStatus method), 241
is_target_is_indirect_member() (dropbox.sharing.PermissionDeniedReason method), 146
is_target_is_owner() (dropbox.sharing.PermissionDeniedReason method), 146
is_target_is_self() (dropbox.sharing.PermissionDeniedReason method), 146
is_target_not_active() (dropbox.sharing.PermissionDeniedReason method), 146
is_team() (dropbox.sharing.MemberPolicy method), 140
is_team() (dropbox.users.SpaceAllocation method), 251
is_team_admin() (dropbox.team.AdminTier method), 183
is_team_and_password() (dropbox.sharing.ResolvedVisibility method), 154
is_team_and_password() (dropbox.sharing.Visibility method), 181
is_team_folder (dropbox.sharing.SharedFolderMetadataBase attribute), 165
is_team_folder() (dropbox.sharing.AddFolderMemberError method), 102
is_team_folder() (dropbox.sharing.RelinquishFolderMembershipError method), 149
is_team_folder() (dropbox.sharing.RemoveFolderMemberError method), 152
is_team_folder() (dropbox.sharing.TransferFolderError method), 173
is_team_folder() (dropbox.sharing.UnshareFolderError method), 176
is_team_license_limit() (dropbox.team.MemberAddResult method), 217
is_team_license_limit() (dropbox.team.MembersRecoverError method), 225
is_team_license_limit() (dropbox.team.MembersSetPermissionsError method), 229
is_team_license_limit() (dropbox.team.MembersSuspendError method), 232
is_team_license_limit() (dropbox.team.MembersUnsuspendError method), 233
is_team_member_id() (dropbox.team.UserSelectorArg method), 243
is_team_member_ids() (dropbox.team.UsersSelectorArg method), 245
is_team_only() (dropbox.sharing.RequestedVisibility method), 153
is_team_only() (dropbox.sharing.SharedLinkAccessFailureReason method), 166
is_team_only() (dropbox.sharing.Visibility method), 181
is_team_policy_disallows_member_policy() (dropbox.sharing.ShareFolderErrorBase method), 156
is_team_policy_disallows_member_policy() (dropbox.sharing.UpdateFolderPolicyError method),

180
is_teammate (dropbox.users.BasicAccount attribute), 247
is_to() (dropbox.files.RelocationError method), 75
is_too_many_files() (dropbox.files.RelocationError method), 75
is_too_many_files() (dropbox.files.SaveCopyReferenceError method), 79
is_too_many_invitees() (dropbox.sharing.AddFolderMemberError method), 102
is_too_many_members() (dropbox.sharing.AddFolderMemberError method), 103
is_too_many_pending_invites() (dropbox.sharing.AddFolderMemberError method), 103
is_too_many_shared_folder_targets() (dropbox.files.UploadSessionFinishError method), 91
is_too_many_write_operations() (dropbox.files.DeleteBatchError method), 51
is_too_many_write_operations() (dropbox.files.RelocationBatchError method), 73
is_transfer_admin_is_not_admin() (dropbox.team.MembersRemoveError method), 228
is_transfer_admin_user_not_found() (dropbox.team.MembersRemoveError method), 228
is_transfer_admin_user_not_in_team() (dropbox.team.MembersRemoveError method), 228
is_transfer_dest_user_not_found() (dropbox.team.MembersRemoveError method), 228
is_transfer_dest_user_not_in_team() (dropbox.team.MembersRemoveError method), 228
is_unmount() (dropbox.sharing.FolderAction method), 116
is_unmounted() (dropbox.sharing.SharedFolderAccessError method), 162
is_unshare() (dropbox.sharing.FileAction method), 109
is_unshare() (dropbox.sharing.FolderAction method), 116
is_unshare_folder_error() (dropbox.sharing.JobError method), 125
is_unspecified_transfer_admin_id() (dropbox.team.MembersRemoveError method), 228
is_unsupported_content() (dropbox.files.PreviewError method), 70
is_unsupported_extension() (dropbox.files.PreviewError method), 71
is_unsupported_extension() (dropbox.files.ThumbnailError method), 86
is_unsupported_image() (dropbox.files.ThumbnailError method), 86
is_unsuspend_non_suspended_member() (dropbox.team.MembersUnsuspendError method), 233
is_unverified_dropbox_id() (dropbox.sharing.AddMemberSelectorError method), 105
is_update() (dropbox.files.WriteMode method), 96
is_user_already_on_team() (dropbox.team.MemberAddResult method), 217
is_user_already_paired() (dropbox.team.MemberAddResult method), 217
is_user_cannot_be_manager_of_company_managed_group() (dropbox.team.GroupMembersAddError method), 197
is_user_cannot_be_manager_of_company_managed_group() (dropbox.team.GroupMemberSetAccessTypeError method), 195
is_user_creation_failed() (dropbox.team.MemberAddResult method), 217
is_user_error() (dropbox.sharing.AddFileMemberError method), 100
is_user_error() (dropbox.sharing.GetFileMetadataError method), 119
is_user_error() (dropbox.sharing.ListFileMembersContinueError method), 129
is_user_error() (dropbox.sharing.ListFileMembersError method), 130
is_user_error() (dropbox.sharing.ListFilesContinueError method), 132
is_user_error() (dropbox.sharing.RemoveFileMemberError method), 150
is_user_error() (dropbox.sharing.UnshareFileError method), 175
is_user_management_admin() (dropbox.team.AdminTier method), 183
is_user_migration_failed() (dropbox.team.MemberAddResult method), 217
is_user_must_be_active_to_be_owner() (dropbox.team.GroupMembersAddError method), 197
is_user_not_allowed_by_owner() (dropbox.sharing.PermissionDeniedReason method), 146
is_user_not_found() (dropbox.team.UserSelectorError method), 244
is_user_not_in_team() (dropbox.team.MembersDeactivateError method), 222

is_user_not_in_team() (dropbox.team.MemberSelectorError method), 220
is_user_not_in_team() (dropbox.team.MembersRecoverError method), 226
is_user_not_in_team() (dropbox.team.MembersSetPermissionsError method), 229
is_user_not_same_team_as_owner() (dropbox.sharing.PermissionDeniedReason method), 146
is_user_on_another_team() (dropbox.team.MemberAddResult method), 217
is_user_unrecoverable() (dropbox.team.MembersRecoverError method), 226
is_users_not_found() (dropbox.team.GroupMembersAddError method), 197
is_viewer() (dropbox.sharing.AccessLevel method), 97
is_viewer_no_comment() (dropbox.sharing.AccessLevel method), 97
is_w1024h768() (dropbox.files.ThumbnailSize method), 86
is_w128h128() (dropbox.files.ThumbnailSize method), 87
is_w32h32() (dropbox.files.ThumbnailSize method), 87
is_w640h480() (dropbox.files.ThumbnailSize method), 87
is_w64h64() (dropbox.files.ThumbnailSize method), 87
is_web_session() (dropbox.team.RevokeDeviceSessionArg method), 236
is_windows() (dropbox.team.DesktopPlatform method), 186
is_windows_phone() (dropbox.team.MobileClientPlatform method), 234

J

JobError (class in dropbox.sharing), 124
JobStatus (class in dropbox.sharing), 125
jpeg (dropbox.files.ThumbnailFormat attribute), 86

K

keep_account (dropbox.team.MembersRemoveArg attribute), 226
keep_app_folder (dropbox.team.RevokeLinkedApiAppArg attribute), 237

L

last_admin (dropbox.team.MembersSetPermissionsError attribute), 229
last_carrier (dropbox.team.MobileClientSession attribute), 234
latitude (dropbox.files.GpsCoordinates attribute), 61
LaunchEmptyResult (class in dropbox.async), 46
LaunchResultBase (class in dropbox.async), 46
leave_a_copy (dropbox.sharing.FolderAction attribute), 116
leave_a_copy (dropbox.sharing.MemberAction attribute), 139
leave_a_copy (dropbox.sharing.RelinquishFolderMembershipArg attribute), 147
leave_a_copy (dropbox.sharing.RemoveFolderMemberArg attribute), 150
leave_a_copy (dropbox.sharing.UnshareFolderArg attribute), 175
licenses (dropbox.team.GetMembershipReport attribute), 191
limit (dropbox.files.ListRevisionsArg attribute), 65
limit (dropbox.sharing.ListFileMembersArg attribute), 128
limit (dropbox.sharing.ListFileMembersBatchArg attribute), 128
limit (dropbox.sharing.ListFilesArg attribute), 132
limit (dropbox.sharing.ListFolderMembersCursorArg attribute), 134
limit (dropbox.sharing.ListFoldersArgs attribute), 135
limit (dropbox.team.GroupsListArg attribute), 202
limit (dropbox.team.GroupsMembersListArg attribute), 204
limit (dropbox.team.MembersListArg attribute), 224
limited (dropbox.team.TeamMembershipType attribute), 242
link (dropbox.files.GetTemporaryLinkResult attribute), 60
link_password (dropbox.sharing.GetSharedLinkMetadataArg attribute), 121
link_password (dropbox.sharing.SharedLinkSettings attribute), 169
link_permissions (dropbox.sharing.SharedLinkMetadata attribute), 168
linked (dropbox.team.ApiApp attribute), 184
linked_api_apps (dropbox.team.ListMemberAppsResult attribute), 207
linked_api_apps (dropbox.team.MemberLinkedApps attribute), 219
LinkMetadata (class in dropbox.sharing), 126
LinkPermissions (class in dropbox.sharing), 126
links (dropbox.sharing.GetSharedLinksResult attribute), 122
links (dropbox.sharing.ListSharedLinksResult attribute), 137

linux (dropbox.team.DesktopPlatform attribute), 186
linux (dropbox.team.DevicesActive attribute), 187
ListFileMembersArg (class in dropbox.sharing), 127
ListFileMembersBatchArg (class in dropbox.sharing), 128
ListFileMembersBatchResult (class in dropbox.sharing), 128
ListFileMembersContinueArg (class in dropbox.sharing), 128
ListFileMembersContinueError (class in dropbox.sharing), 129
ListFileMembersCountResult (class in dropbox.sharing), 129
ListFileMembersError (class in dropbox.sharing), 130
ListFileMembersIndividualResult (class in dropbox.sharing), 130
ListFilesArg (class in dropbox.sharing), 131
ListFilesContinueArg (class in dropbox.sharing), 132
ListFilesContinueError (class in dropbox.sharing), 132
ListFilesResult (class in dropbox.sharing), 133
ListFolderArg (class in dropbox.files), 61
ListFolderContinueArg (class in dropbox.files), 62
ListFolderContinueError (class in dropbox.files), 62
ListFolderError (class in dropbox.files), 63
ListFolderGetLatestCursorResult (class in dropbox.files), 63
ListFolderLongpollArg (class in dropbox.files), 63
ListFolderLongpollError (class in dropbox.files), 64
ListFolderLongpollResult (class in dropbox.files), 64
ListFolderMembersArgs (class in dropbox.sharing), 133
ListFolderMembersContinueArg (class in dropbox.sharing), 133
ListFolderMembersContinueError (class in dropbox.sharing), 133
ListFolderMembersCursorArg (class in dropbox.sharing), 134
ListFolderResult (class in dropbox.files), 65
ListFoldersArgs (class in dropbox.sharing), 134
ListFoldersContinueArg (class in dropbox.sharing), 135
ListFoldersContinueError (class in dropbox.sharing), 135
ListFoldersResult (class in dropbox.sharing), 135
ListMemberAppsArg (class in dropbox.team), 206
ListMemberAppsError (class in dropbox.team), 206
ListMemberAppsResult (class in dropbox.team), 206
ListMemberDevicesArg (class in dropbox.team), 207
ListMemberDevicesError (class in dropbox.team), 207
ListMemberDevicesResult (class in dropbox.team), 208
ListMembersAppsArg (class in dropbox.team), 208
ListMembersAppsError (class in dropbox.team), 208
ListMembersAppsResult (class in dropbox.team), 209
ListMembersDevicesArg (class in dropbox.team), 209
ListMembersDevicesError (class in dropbox.team), 210
ListMembersDevicesResult (class in dropbox.team), 210
ListRevisionsArg (class in dropbox.files), 65
ListRevisionsError (class in dropbox.files), 65
ListRevisionsResult (class in dropbox.files), 66
ListSharedLinksArg (class in dropbox.sharing), 136
ListSharedLinksError (class in dropbox.sharing), 136
ListSharedLinksResult (class in dropbox.sharing), 137
ListTeamAppsArg (class in dropbox.team), 211
ListTeamAppsError (class in dropbox.team), 211
ListTeamAppsResult (class in dropbox.team), 212
ListTeamDevicesArg (class in dropbox.team), 212
ListTeamDevicesError (class in dropbox.team), 213
ListTeamDevicesResult (class in dropbox.team), 213
locale (dropbox.users.FullAccount attribute), 247
location (dropbox.files.MediaMetadata attribute), 68
login_required (dropbox.sharing.SharedLinkAccessFailureReason attribute), 166
longitude (dropbox.files.GpsCoordinates attribute), 61
lookup_failed() (dropbox.files.UploadSessionFinishError class method), 92
LookupError (class in dropbox.files), 66
LookUpPropertiesError (class in dropbox.files), 66

M

mac (dropbox.team.DesktopPlatform attribute), 186
macos (dropbox.team.DevicesActive attribute), 187
make_editor (dropbox.sharing.MemberAction attribute), 139
make_owner (dropbox.sharing.MemberAction attribute), 139
make_viewer (dropbox.sharing.MemberAction attribute), 139
make_viewer_no_comment (dropbox.sharing.MemberAction attribute), 139
malformed_path() (dropbox.files.LookupError class method), 67
malformed_path() (dropbox.files.WriteError class method), 96
match_type (dropbox.files.SearchMatch attribute), 83
matches (dropbox.files.SearchResult attribute), 84
max_results (dropbox.files.SearchArg attribute), 82
media_info (dropbox.files.FileMetadata attribute), 56
MediaInfo (class in dropbox.files), 68
MediaMetadata (class in dropbox.files), 68
member (dropbox.sharing.AddMember attribute), 103
member (dropbox.sharing.ChangeFileMemberAccessArgs attribute), 105
member (dropbox.sharing.FileMemberActionResult attribute), 113
member (dropbox.sharing.RemoveFileMemberArg attribute), 149
member (dropbox.sharing.RemoveFolderMemberArg attribute), 151
member (dropbox.sharing.UpdateFolderMemberArg attribute), 176
member (dropbox.team.GroupAccessType attribute), 192

member_count (dropbox.sharing.ListFileMembersCountResult attribute), 130
member_email (dropbox.team.MemberAddArg attribute), 214
member_error() (dropbox.sharing.FileMemberActionIndividualResult class method), 112
member_error() (dropbox.sharing.FileMemberRemoveActionResult class method), 114
member_error() (dropbox.sharing.RemoveFolderMemberError class method), 152
member_error() (dropbox.sharing.UpdateFolderMemberError class method), 178
member_external_id (dropbox.team.MemberAddArg attribute), 214
member_given_name (dropbox.team.MemberAddArg attribute), 214
member_id (dropbox.sharing.TeamMemberInfo attribute), 171
member_info() (dropbox.team.MembersGetInfoItem class method), 223
member_not_found (dropbox.team.ListMemberAppsError attribute), 206
member_not_found (dropbox.team.ListMemberDevicesError attribute), 208
member_not_found (dropbox.team.RevokeDeviceSessionError attribute), 237
member_not_found (dropbox.team.RevokeLinkedAppError attribute), 238
member_not_in_group (dropbox.team.GroupMemberSelectorError attribute), 195
member_not_in_group (dropbox.team.GroupMembersSelectorError attribute), 199
member_only (dropbox.team.AdminTier attribute), 183
member_policy (dropbox.sharing.FolderPolicy attribute), 117
member_policy (dropbox.sharing.ShareFolderArg attribute), 155
member_policy (dropbox.sharing.UpdateFolderPolicyArg attribute), 178
member_storage_map (dropbox.team.GetStorageReport attribute), 192
member_surname (dropbox.team.MemberAddArg attribute), 214
MemberAccess (class in dropbox.team), 214
MemberAccessLevelResult (class in dropbox.sharing), 137
MemberAction (class in dropbox.sharing), 138
MemberAddArg (class in dropbox.team), 214
MemberAddResult (class in dropbox.team), 215
MemberDevices (class in dropbox.team), 218
MemberLinkedApps (class in dropbox.team), 218
MemberPermission (class in dropbox.sharing), 139
MemberPolicy (class in dropbox.sharing), 139
MemberProfile (class in dropbox.team), 219
members (dropbox.sharing.AddFileMemberArgs attribute), 99
members (dropbox.sharing.AddFolderMemberArg attribute), 100
members (dropbox.sharing.ListFileMembersCountResult attribute), 130
members (dropbox.sharing.SharedLinkPolicy attribute), 169
members (dropbox.team.GroupFullInfo attribute), 194
members (dropbox.team.GroupMembersAddArg attribute), 195
members (dropbox.team.GroupsMembersListResult attribute), 205
members (dropbox.team.MembersGetInfoArgs attribute), 222
members (dropbox.team.MembersListResult attribute), 225
members_joined (dropbox.team.GetMembershipReport attribute), 191
members_not_in_team() (dropbox.team.GroupMembersAddError class method), 197
MembersAddArg (class in dropbox.team), 220
MembersAddJobStatus (class in dropbox.team), 220
MembersAddLaunch (class in dropbox.team), 221
MembersDeactivateArg (class in dropbox.team), 222
MembersDeactivateError (class in dropbox.team), 222
MemberSelector (class in dropbox.sharing), 140
MemberSelectorError (class in dropbox.team), 220
MembersGetInfoArgs (class in dropbox.team), 222
MembersGetInfoError (class in dropbox.team), 222
MembersGetInfoItem (class in dropbox.team), 223
membership_type (dropbox.team.MemberProfile attribute), 219
MembershipInfo (class in dropbox.sharing), 141
MembersListArg (class in dropbox.team), 223
MembersListContinueArg (class in dropbox.team), 224
MembersListContinueError (class in dropbox.team), 224
MembersListError (class in dropbox.team), 224
MembersListResult (class in dropbox.team), 224
MembersRecoverArg (class in dropbox.team), 225
MembersRecoverError (class in dropbox.team), 225
MembersRemoveArg (class in dropbox.team), 226
MembersRemoveError (class in dropbox.team), 226
MembersSendWelcomeError (class in dropbox.team), 228
MembersSetPermissionsArg (class in dropbox.team), 228

MembersSetPermissionsError (class in dropbox.team), 229
MembersSetPermissionsResult (class in dropbox.team), 230
MembersSetProfileArg (class in dropbox.team), 230
MembersSetProfileError (class in dropbox.team), 231
MembersSuspendError (class in dropbox.team), 232
MembersUnsuspendArg (class in dropbox.team), 232
MembersUnsuspendError (class in dropbox.team), 232
Metadata (class in dropbox.files), 69
metadata (dropbox.files.DeleteResult attribute), 54
metadata (dropbox.files.GetCopyReferenceResult attribute), 59
metadata (dropbox.files.GetTemporaryLinkResult attribute), 60
metadata (dropbox.files.RelocationResult attribute), 76
metadata (dropbox.files.SaveCopyReferenceResult attribute), 79
metadata (dropbox.files.SearchMatch attribute), 83
metadata() (dropbox.files.MediaInfo class method), 68
metadata() (dropbox.sharing.GetFileMetadataIndividualResult class method), 120
mobile_client() (dropbox.team.RevokeDeviceSessionArg class method), 236
mobile_client_sessions (dropbox.team.ListMemberDevicesResult attribute), 208
mobile_clients (dropbox.team.MemberDevices attribute), 218
MobileClientPlatform (class in dropbox.team), 233
MobileClientSession (class in dropbox.team), 234
mode (dropbox.files.CommitInfo attribute), 50
mode (dropbox.files.SearchArg attribute), 82
modified_by (dropbox.files.FileSharingInfo attribute), 56
ModifySharedLinkSettingsArgs (class in dropbox.sharing), 141
ModifySharedLinkSettingsError (class in dropbox.sharing), 142
more (dropbox.files.SearchResult attribute), 84
mounted (dropbox.sharing.RelinquishFolderMembershipError attribute), 149
MountFolderArg (class in dropbox.sharing), 142
MountFolderError (class in dropbox.sharing), 143
mute (dropbox.files.CommitInfo attribute), 50

N

Name (class in dropbox.users), 249
name (dropbox.files.Metadata attribute), 69
name (dropbox.sharing.SharedFileMetadata attribute), 161
name (dropbox.sharing.SharedFolderMetadata attribute), 165
name (dropbox.sharing.SharedLinkMetadata attribute), 168
name (dropbox.team.MemberProfile attribute), 220
name (dropbox.team.TeamGetInfoResult attribute), 239
name (dropbox.team.UpdatePropertyTemplateArg attribute), 242
name (dropbox.users.Account attribute), 246
name (dropbox.users.Team attribute), 252
new_email (dropbox.team.MembersSetProfileArg attribute), 230
new_external_id (dropbox.team.MembersSetProfileArg attribute), 230
new_given_name (dropbox.team.MembersSetProfileArg attribute), 230
new_group_external_id (dropbox.team.GroupUpdateArgs attribute), 201
new_group_management_type (dropbox.team.GroupUpdateArgs attribute), 201
new_group_name (dropbox.team.GroupUpdateArgs attribute), 201
new_members (dropbox.team.MembersAddArg attribute), 220
new_owner_email_unverified (dropbox.sharing.TransferFolderError attribute), 173
new_owner_not_a_member (dropbox.sharing.TransferFolderError attribute), 173
new_owner_unmounted (dropbox.sharing.TransferFolderError attribute), 173
new_role (dropbox.team.MembersSetPermissionsArg attribute), 229
new_surname (dropbox.team.MembersSetProfileArg attribute), 230
no_access (dropbox.files.FolderSharingInfo attribute), 57
no_account (dropbox.users.GetAccountError attribute), 249
no_account() (dropbox.users.GetAccountBatchError class method), 249
no_explicit_access() (dropbox.sharing.RemoveFileMemberError class method), 150
no_explicit_access() (dropbox.sharing.SharedFolderMemberError class method), 163
no_explicit_access() (dropbox.sharing.UpdateFolderMemberError class method), 178
no_new_data_specified (dropbox.team.MembersSetProfileError attribute), 232
no_permission (dropbox.files.SaveCopyReferenceError attribute), 79
no_permission (dropbox.sharing.AddFolderMemberError attribute), 103

no_permission (dropbox.sharing.FileMemberActionError attribute), 112
no_permission (dropbox.sharing.MountFolderError attribute), 144
no_permission (dropbox.sharing.RelinquishFileMembershipError attribute), 147
no_permission (dropbox.sharing.RelinquishFolderMembershipError attribute), 149
no_permission (dropbox.sharing.RemoveFolderMemberError attribute), 152
no_permission (dropbox.sharing.ShareFolderError attribute), 155
no_permission (dropbox.sharing.SharingFileAccessError attribute), 171
no_permission (dropbox.sharing.TransferFolderError attribute), 173
no_permission (dropbox.sharing.UnmountFolderError attribute), 174
no_permission (dropbox.sharing.UnshareFolderError attribute), 176
no_permission (dropbox.sharing.UpdateFolderMemberError attribute), 178
no_permission (dropbox.sharing.UpdateFolderPolicyError attribute), 180
no_write_permission (dropbox.files.WriteError attribute), 96
not_a_member (dropbox.sharing.SharedFolderAccessError attribute), 163
not_a_member (dropbox.sharing.SharedFolderMemberError attribute), 163
not_authorized (dropbox.sharing.SharedLinkSettingsError attribute), 170
not_closed (dropbox.files.UploadSessionLookupError attribute), 93
not_file (dropbox.files.LookupError attribute), 68
not_folder (dropbox.files.LookupError attribute), 68
not_found (dropbox.files.LookupError attribute), 68
not_found (dropbox.files.SaveCopyReferenceError attribute), 79
not_found (dropbox.files.SaveUrlError attribute), 80
not_found (dropbox.files.UploadSessionLookupError attribute), 93
not_mountable (dropbox.sharing.MountFolderError attribute), 144
not_on_team (dropbox.sharing.UpdateFolderPolicyError attribute), 180
not_unmountable (dropbox.sharing.UnmountFolderError attribute), 174
NotApprovedException, 5
num_licensed_users (dropbox.team.TeamGetInfoResult attribute), 240
num_provisioned_users (dropbox.team.TeamGetInfoResult attribute), 240

O

OAuth2FlowNoRedirectResult (class in dropbox.oauth), 5
OAuth2FlowResult (class in dropbox.oauth), 5
offset (dropbox.files.UploadSessionCursor attribute), 89
os (dropbox.team.ActiveWebSession attribute), 182
os_version (dropbox.team.MobileClientSession attribute), 234
other (dropbox.async.PollError attribute), 48
other (dropbox.files.DeleteBatchError attribute), 51
other (dropbox.files.DeleteBatchJobStatus attribute), 52
other (dropbox.files.DeleteError attribute), 53
other (dropbox.files.DownloadError attribute), 55
other (dropbox.files.GetCopyReferenceError attribute), 58
other (dropbox.files.GetTemporaryLinkError attribute), 60
other (dropbox.files.ListFolderContinueError attribute), 63
other (dropbox.files.ListFolderError attribute), 63
other (dropbox.files.ListFolderLongpollError attribute), 64
other (dropbox.files.ListRevisionsError attribute), 66
other (dropbox.files.LookupError attribute), 68
other (dropbox.files.RelocationError attribute), 75
other (dropbox.files.RestoreError attribute), 78
other (dropbox.files.SaveCopyReferenceError attribute), 79
other (dropbox.files.SaveUrlError attribute), 80
other (dropbox.files.SearchError attribute), 83
other (dropbox.files.UploadError attribute), 88
other (dropbox.files.UploadSessionFinishError attribute), 92
other (dropbox.files.UploadSessionLookupError attribute), 93
other (dropbox.files.WriteConflictError attribute), 94
other (dropbox.files.WriteError attribute), 96
other (dropbox.sharing.AccessLevel attribute), 97
other (dropbox.sharing.AclUpdatePolicy attribute), 98
other (dropbox.sharing.AddFileMemberError attribute), 100
other (dropbox.sharing.AddFolderMemberError attribute), 103
other (dropbox.sharing.AddMemberSelectorError attribute), 105
other (dropbox.sharing.CreateSharedLinkError attribute), 106
other (dropbox.sharing.FileAction attribute), 109
other (dropbox.sharing.FileErrorResult attribute), 110
other (dropbox.sharing.FileMemberActionError attribute), 112
other (dropbox.sharing.FileMemberRemoveActionResult attribute), 114
other (dropbox.sharing.FolderAction attribute), 116

other (dropbox.sharing.GetFileMetadataError attribute), 119
other (dropbox.sharing.GetFileMetadataIndividualResult attribute), 120
other (dropbox.sharing.GetSharedLinksError attribute), 122
other (dropbox.sharing.InviteeInfo attribute), 124
other (dropbox.sharing.JobError attribute), 125
other (dropbox.sharing.ListFileMembersContinueError attribute), 129
other (dropbox.sharing.ListFileMembersError attribute), 130
other (dropbox.sharing.ListFileMembersIndividualResult attribute), 131
other (dropbox.sharing.ListFilesContinueError attribute), 132
other (dropbox.sharing.ListFolderMembersContinueError attribute), 134
other (dropbox.sharing.ListFoldersContinueError attribute), 135
other (dropbox.sharing.ListSharedLinksError attribute), 137
other (dropbox.sharing.MemberAction attribute), 139
other (dropbox.sharing.MemberPolicy attribute), 140
other (dropbox.sharing.MemberSelector attribute), 141
other (dropbox.sharing.MountFolderError attribute), 144
other (dropbox.sharing.PermissionDeniedReason attribute), 146
other (dropbox.sharing.RelinquishFileMembershipError attribute), 147
other (dropbox.sharing.RelinquishFolderMembershipError attribute), 149
other (dropbox.sharing.RemoveFileMemberError attribute), 150
other (dropbox.sharing.RemoveFolderMemberError attribute), 152
other (dropbox.sharing.ResolvedVisibility attribute), 154
other (dropbox.sharing.SharedFolderAccessError attribute), 163
other (dropbox.sharing.SharedFolderMemberError attribute), 163
other (dropbox.sharing.SharedLinkAccessFailureReason attribute), 167
other (dropbox.sharing.SharedLinkError attribute), 167
other (dropbox.sharing.SharedLinkPolicy attribute), 169
other (dropbox.sharing.ShareFolderErrorBase attribute), 156
other (dropbox.sharing.SharePathError attribute), 160
other (dropbox.sharing.SharingFileAccessError attribute), 171
other (dropbox.sharing.SharingUserError attribute), 171
other (dropbox.sharing.TransferFolderError attribute), 173
other (dropbox.sharing.UnmountFolderError attribute), 174
other (dropbox.sharing.UnshareFileError attribute), 175
other (dropbox.sharing.UnshareFolderError attribute), 176
other (dropbox.sharing.UpdateFolderMemberError attribute), 178
other (dropbox.sharing.UpdateFolderPolicyError attribute), 180
other (dropbox.sharing.Visibility attribute), 181
other (dropbox.team.DateRangeError attribute), 185
other (dropbox.team.DesktopPlatform attribute), 186
other (dropbox.team.DevicesActive attribute), 188
other (dropbox.team.GroupCreateError attribute), 193
other (dropbox.team.GroupSelectorError attribute), 200
other (dropbox.team.GroupsGetInfoError attribute), 201
other (dropbox.team.GroupsListContinueError attribute), 203
other (dropbox.team.GroupsMembersListContinueError attribute), 204
other (dropbox.team.ListMemberAppsError attribute), 206
other (dropbox.team.ListMemberDevicesError attribute), 208
other (dropbox.team.ListMembersAppsError attribute), 209
other (dropbox.team.ListMembersDevicesError attribute), 210
other (dropbox.team.ListTeamAppsError attribute), 211
other (dropbox.team.ListTeamDevicesError attribute), 213
other (dropbox.team.MembersDeactivateError attribute), 222
other (dropbox.team.MembersGetInfoError attribute), 223
other (dropbox.team.MembersListContinueError attribute), 224
other (dropbox.team.MembersListError attribute), 224
other (dropbox.team.MembersRecoverError attribute), 226
other (dropbox.team.MembersSendWelcomeError attribute), 228
other (dropbox.team.MembersSetPermissionsError attribute), 229
other (dropbox.team.MembersSetProfileError attribute), 232
other (dropbox.team.MobileClientPlatform attribute), 234
other (dropbox.team.RevokeDeviceSessionBatchError attribute), 236
other (dropbox.team.RevokeDeviceSessionError attribute), 237
other (dropbox.team.RevokeLinkedAppBatchError attribute), 238

other (dropbox.team.RevokeLinkedAppError attribute), 238
other (dropbox.users.GetAccountBatchError attribute), 249
other (dropbox.users.GetAccountError attribute), 249
other (dropbox.users.SpaceAllocation attribute), 251
overwrite (dropbox.files.WriteMode attribute), 97
owner (dropbox.sharing.AccessLevel attribute), 97
owner (dropbox.sharing.AclUpdatePolicy attribute), 98
owner (dropbox.team.GroupAccessType attribute), 192
owner_only (dropbox.sharing.SharedLinkAccessFailureReason attribute), 167
owner_team (dropbox.sharing.SharedFileMetadata attribute), 161
owner_team (dropbox.sharing.SharedFolderMetadataBase attribute), 166

P

param_cannot_be_empty (dropbox.team.MembersSetProfileError attribute), 232
parent_shared_folder_id (dropbox.files.FileSharingInfo attribute), 56
parent_shared_folder_id (dropbox.files.FolderSharingInfo attribute), 57
parent_shared_folder_id (dropbox.files.Metadata attribute), 69
parent_shared_folder_id (dropbox.sharing.SharedFileMetadata attribute), 161
parent_shared_folder_id (dropbox.sharing.SharedFolderMetadataBase attribute), 166
ParentFolderAccessInfo (class in dropbox.sharing), 144
password (dropbox.sharing.RequestedVisibility attribute), 153
password (dropbox.sharing.Visibility attribute), 181
password_required (dropbox.sharing.SharedLinkAccessFailureReason attribute), 167
path (dropbox.files.CommitInfo attribute), 50
path (dropbox.files.CreateFolderArg attribute), 50
path (dropbox.files.DeleteArg attribute), 51
path (dropbox.files.DownloadArg attribute), 54
path (dropbox.files.GetCopyReferenceArg attribute), 58
path (dropbox.files.GetMetadataArg attribute), 59
path (dropbox.files.GetTemporaryLinkArg attribute), 60
path (dropbox.files.ListFolderArg attribute), 62
path (dropbox.files.ListRevisionsArg attribute), 65
path (dropbox.files.PreviewArg attribute), 70
path (dropbox.files.PropertyGroupWithPath attribute), 72
path (dropbox.files.RemovePropertiesArg attribute), 76
path (dropbox.files.RestoreArg attribute), 77
path (dropbox.files.SaveCopyReferenceArg attribute), 78
path (dropbox.files.SaveUrlArg attribute), 79
path (dropbox.files.SearchArg attribute), 82
path (dropbox.files.ThumbnailArg attribute), 85
path (dropbox.files.UpdatePropertyGroupArg attribute), 87
path (dropbox.sharing.CreateSharedLinkArg attribute), 106
path (dropbox.sharing.CreateSharedLinkWithSettingsArg attribute), 107
path (dropbox.sharing.GetSharedLinkMetadataArg attribute), 121
path (dropbox.sharing.GetSharedLinksArg attribute), 121
path (dropbox.sharing.ListSharedLinksArg attribute), 136
path (dropbox.sharing.PathLinkMetadata attribute), 145
path (dropbox.sharing.ShareFolderArg attribute), 155
path() (dropbox.files.CreateFolderError class method), 51
path() (dropbox.files.DownloadError class method), 55
path() (dropbox.files.GetCopyReferenceError class method), 58
path() (dropbox.files.GetMetadataError class method), 60
path() (dropbox.files.GetTemporaryLinkError class method), 60
path() (dropbox.files.ListFolderContinueError class method), 63
path() (dropbox.files.ListFolderError class method), 63
path() (dropbox.files.ListRevisionsError class method), 66
path() (dropbox.files.PreviewError class method), 71
path() (dropbox.files.PropertiesError class method), 71
path() (dropbox.files.SaveCopyReferenceError class method), 79
path() (dropbox.files.SaveUrlError class method), 80
path() (dropbox.files.SearchError class method), 83
path() (dropbox.files.ThumbnailError class method), 86
path() (dropbox.files.UploadError class method), 88
path() (dropbox.files.UploadSessionFinishError class method), 92
path() (dropbox.sharing.CreateSharedLinkError class method), 106
path() (dropbox.sharing.CreateSharedLinkWithSettingsError class method), 108
path() (dropbox.sharing.GetSharedLinksError class method), 122
path() (dropbox.sharing.ListSharedLinksError class method), 137
path_display (dropbox.files.Metadata attribute), 69
path_display (dropbox.sharing.SharedFileMetadata attribute), 161
path_lookup() (dropbox.files.DeleteError class method), 53
path_lookup() (dropbox.files.RestoreError class method), 78
path_lower (dropbox.files.Metadata attribute), 69

path_lower (dropbox.sharing.SharedFileMetadata attribute), 161
path_lower (dropbox.sharing.SharedFolderMetadata attribute), 165
path_lower (dropbox.sharing.SharedLinkMetadata attribute), 168
path_root (dropbox.files.PathRootError attribute), 69
path_write() (dropbox.files.DeleteError class method), 53
path_write() (dropbox.files.RestoreError class method), 78
PathLinkMetadata (class in dropbox.sharing), 144
PathRootError (class in dropbox.files), 69
pending (dropbox.files.MediaInfo attribute), 68
pending_invites (dropbox.team.GetMembershipReport attribute), 191
pending_upload (dropbox.sharing.CreateSharedLinkArg attribute), 106
PendingUploadMode (class in dropbox.sharing), 145
permission_denied_error() (dropbox.sharing.FileErrorResult class method), 110
PermissionDeniedReason (class in dropbox.sharing), 145
permissions (dropbox.sharing.MembershipInfo attribute), 141
permissions (dropbox.sharing.ParentFolderAccessInfo attribute), 144
permissions (dropbox.sharing.SharedFileMetadata attribute), 161
permissions (dropbox.sharing.SharedFolderMetadata attribute), 165
PhotoMetadata (class in dropbox.files), 70
platform (dropbox.team.DesktopClientSession attribute), 185
png (dropbox.files.ThumbnailFormat attribute), 86
policies (dropbox.team.TeamGetInfoResult attribute), 240
policy (dropbox.sharing.SharedFileMetadata attribute), 162
policy (dropbox.sharing.SharedFolderMetadataBase attribute), 166
PollArg (class in dropbox.async), 47
PollEmptyResult (class in dropbox.async), 47
PollError (class in dropbox.async), 47
PollResultBase (class in dropbox.async), 48
preview_url (dropbox.sharing.SharedFileMetadata attribute), 162
preview_url (dropbox.sharing.SharedFolderMetadata attribute), 165
PreviewArg (class in dropbox.files), 70
PreviewError (class in dropbox.files), 70
pro (dropbox.users.AccountType attribute), 246
profile (dropbox.team.GroupMemberInfo attribute), 194
profile (dropbox.team.TeamMemberInfo attribute), 240
profile_photo_url (dropbox.users.Account attribute), 246
properties_error() (dropbox.files.AlphaGetMetadataError class method), 49
properties_error() (dropbox.files.UploadErrorWithProperties class method), 88
PropertiesError (class in dropbox.files), 71
property_field_too_large (dropbox.files.InvalidPropertyGroupError attribute), 61
property_group_already_exists (dropbox.files.AddPropertiesError attribute), 48
property_group_lookup() (dropbox.files.RemovePropertiesError class method), 76
property_group_lookup() (dropbox.files.UpdatePropertiesError class method), 87
property_group_not_found (dropbox.files.LookUpPropertiesError attribute), 66
property_groups (dropbox.files.CommitInfoWithProperties attribute), 50
property_groups (dropbox.files.FileMetadata attribute), 56
property_groups (dropbox.files.FolderMetadata attribute), 57
property_groups (dropbox.files.PropertyGroupWithPath attribute), 72
property_template_ids (dropbox.files.RemovePropertiesArg attribute), 76
PropertyGroupUpdate (class in dropbox.files), 71
PropertyGroupWithPath (class in dropbox.files), 72
ProviderException, 5
public (dropbox.sharing.RequestedVisibility attribute), 153
public (dropbox.sharing.Visibility attribute), 181
publisher (dropbox.team.ApiApp attribute), 184
publisher_url (dropbox.team.ApiApp attribute), 184

Q

query (dropbox.files.SearchArg attribute), 82
quiet (dropbox.sharing.AddFileMemberArgs attribute), 99
quiet (dropbox.sharing.AddFolderMemberArg attribute), 100

R

rate_limit (dropbox.sharing.AddFileMemberError attribute), 100
rate_limit (dropbox.sharing.AddFolderMemberError attribute), 103
RateLimitError, 253

read_only (dropbox.files.SharingInfo attribute), 85
reason (dropbox.files.UploadWriteFailed attribute), 93
reason (dropbox.sharing.FilePermission attribute), 114
reason (dropbox.sharing.FolderPermission attribute), 117
reason (dropbox.sharing.MemberPermission attribute), 139
recursive (dropbox.files.ListFolderArg attribute), 62
referral_link (dropbox.users.FullAccount attribute), 248
relinquish_folder_membership_error() (dropbox.sharing.JobError class method), 125
relinquish_membership (dropbox.sharing.FileAction attribute), 109
relinquish_membership (dropbox.sharing.FolderAction attribute), 116
RelinquishFileMembershipArg (class in dropbox.sharing), 146
RelinquishFileMembershipError (class in dropbox.sharing), 146
RelinquishFolderMembershipArg (class in dropbox.sharing), 147
RelinquishFolderMembershipError (class in dropbox.sharing), 148
RelocationArg (class in dropbox.files), 72
RelocationBatchArg (class in dropbox.files), 72
RelocationBatchError (class in dropbox.files), 73
RelocationBatchJobStatus (class in dropbox.files), 73
RelocationBatchResult (class in dropbox.files), 74
RelocationError (class in dropbox.files), 74
RelocationPath (class in dropbox.files), 76
RelocationResult (class in dropbox.files), 76
remove (dropbox.sharing.MemberAction attribute), 139
remove_expiration (dropbox.sharing.ModifySharedLinkSettingsArgs attribute), 142
remove_fields (dropbox.files.PropertyGroupUpdate attribute), 71
remove_folder_member_error() (dropbox.sharing.JobError class method), 125
remove_last_admin (dropbox.team.MembersRemoveError attribute), 228
removed() (dropbox.team.TeamMemberStatus class method), 241
removed_and_transfer_admin_should_differ (dropbox.team.MembersRemoveError attribute), 228
removed_and_transfer_dest_should_differ (dropbox.team.MembersRemoveError attribute), 228
RemovedStatus (class in dropbox.team), 234
RemoveFileMemberArg (class in dropbox.sharing), 149
RemoveFileMemberError (class in dropbox.sharing), 149
RemoveFolderMemberArg (class in dropbox.sharing), 150
RemoveFolderMemberError (class in dropbox.sharing), 151
RemoveMemberJobStatus (class in dropbox.sharing), 152
RemovePropertiesArg (class in dropbox.files), 76
RemovePropertiesError (class in dropbox.files), 76
request() (dropbox.dropbox.Dropbox method), 19
request() (dropbox.dropbox.DropboxTeam method), 33
request_json_object() (dropbox.dropbox.Dropbox method), 19
request_json_object() (dropbox.dropbox.DropboxTeam method), 33
request_json_string() (dropbox.dropbox.Dropbox method), 20
request_json_string() (dropbox.dropbox.DropboxTeam method), 34
request_json_string_with_retry() (dropbox.dropbox.Dropbox method), 20
request_json_string_with_retry() (dropbox.dropbox.DropboxTeam method), 34
requested_visibility (dropbox.sharing.LinkPermissions attribute), 127
requested_visibility (dropbox.sharing.SharedLinkSettings attribute), 169
RequestedVisibility (class in dropbox.sharing), 153
reset (dropbox.files.ListFolderContinueError attribute), 63
reset (dropbox.files.ListFolderLongpollError attribute), 64
reset (dropbox.sharing.ListSharedLinksError attribute), 137
reset (dropbox.team.ListMembersAppsError attribute), 209
reset (dropbox.team.ListMembersDevicesError attribute), 210
reset (dropbox.team.ListTeamAppsError attribute), 211
reset (dropbox.team.ListTeamDevicesError attribute), 213
resolved_member_policy (dropbox.sharing.FolderPolicy attribute), 117
resolved_visibility (dropbox.sharing.LinkPermissions attribute), 127
ResolvedVisibility (class in dropbox.sharing), 153
RestoreArg (class in dropbox.files), 77
RestoreError (class in dropbox.files), 77
restricted_content (dropbox.files.LookupError attribute), 68
result (dropbox.sharing.FileMemberActionResult attribute), 113
result (dropbox.sharing.GetFileMetadataBatchResult attribute), 118
result (dropbox.sharing.ListFileMembersBatchResult attribute), 128

result() (dropbox.sharing.ListFileMembersIndividualResult class method), 131
return_members (dropbox.team.GroupMembersSetAccessTypeArg attribute), 199
return_members (dropbox.team.IncludeMembersArg attribute), 206
rev (dropbox.files.DownloadArg attribute), 54
rev (dropbox.files.FileMetadata attribute), 56
rev (dropbox.files.PreviewArg attribute), 70
rev (dropbox.files.RestoreArg attribute), 77
rev (dropbox.sharing.FileLinkMetadata attribute), 111
revoke_devices (dropbox.team.RevokeDeviceSessionBatchArg attribute), 236
revoke_devices_status (dropbox.team.RevokeDeviceSessionBatchResult attribute), 236
revoke_failure_reason (dropbox.sharing.LinkPermissions attribute), 127
revoke_linked_app (dropbox.team.RevokeLinkedApiAppBatchArg attribute), 238
revoke_linked_app_status (dropbox.team.RevokeLinkedAppBatchResult attribute), 238
RevokeDesktopClientArg (class in dropbox.team), 235
RevokeDeviceSessionArg (class in dropbox.team), 235
RevokeDeviceSessionBatchArg (class in dropbox.team), 236
RevokeDeviceSessionBatchError (class in dropbox.team), 236
RevokeDeviceSessionBatchResult (class in dropbox.team), 236
RevokeDeviceSessionError (class in dropbox.team), 236
RevokeDeviceSessionStatus (class in dropbox.team), 237
RevokeLinkedApiAppArg (class in dropbox.team), 237
RevokeLinkedApiAppBatchArg (class in dropbox.team), 238
RevokeLinkedAppBatchError (class in dropbox.team), 238
RevokeLinkedAppBatchResult (class in dropbox.team), 238
RevokeLinkedAppError (class in dropbox.team), 238
RevokeLinkedAppStatus (class in dropbox.team), 239
RevokeSharedLinkArg (class in dropbox.sharing), 154
RevokeSharedLinkError (class in dropbox.sharing), 154
role (dropbox.team.MemberAddArg attribute), 215
role (dropbox.team.MembersSetPermissionsResult attribute), 230
role (dropbox.team.TeamMemberInfo attribute), 240

S

same_team (dropbox.sharing.GroupInfo attribute), 122
same_team (dropbox.sharing.UserInfo attribute), 180
SaveCopyReferenceArg (class in dropbox.files), 78
SaveCopyReferenceError (class in dropbox.files), 78
SaveCopyReferenceResult (class in dropbox.files), 79
SaveUrlArg (class in dropbox.files), 79
SaveUrlError (class in dropbox.files), 80
SaveUrlJobStatus (class in dropbox.files), 80
SaveUrlResult (class in dropbox.files), 81
SearchArg (class in dropbox.files), 82
SearchError (class in dropbox.files), 82
SearchMatch (class in dropbox.files), 83
SearchMatchType (class in dropbox.files), 83
SearchMode (class in dropbox.files), 84
SearchResult (class in dropbox.files), 84
send_welcome_email (dropbox.team.MemberAddArg attribute), 215
server_modified (dropbox.files.FileMetadata attribute), 56
server_modified (dropbox.sharing.FileLinkMetadata attribute), 111
session_id (dropbox.files.UploadSessionCursor attribute), 89
session_id (dropbox.files.UploadSessionStartResult attribute), 93
session_id (dropbox.team.DeviceSession attribute), 186
session_id (dropbox.team.DeviceSessionArg attribute), 187
set_profile_disallowed (dropbox.team.MembersSetProfileError attribute), 232
settings (dropbox.sharing.CreateSharedLinkWithSettingsArg attribute), 107
settings (dropbox.sharing.ModifySharedLinkSettingsArgs attribute), 142
settings_error() (dropbox.sharing.CreateSharedLinkWithSettingsError class method), 108
settings_error() (dropbox.sharing.ModifySharedLinkSettingsError class method), 142
share_link (dropbox.sharing.FileAction attribute), 109
share_link (dropbox.sharing.FolderAction attribute), 116
shared_folder_id (dropbox.files.FolderMetadata attribute), 57
shared_folder_id (dropbox.files.FolderSharingInfo attribute), 58
shared_folder_id (dropbox.sharing.AddFolderMemberArg attribute), 101
shared_folder_id (dropbox.sharing.GetMetadataArgs attribute), 120
shared_folder_id (dropbox.sharing.ListFolderMembersArgs attribute), 133
shared_folder_id (dropbox.sharing.MountFolderArg attribute), 143

shared_folder_id (dropbox.sharing.ParentFolderAccessInfo attribute), 144
shared_folder_id (dropbox.sharing.RelinquishFolderMembershipArg attribute), 147
shared_folder_id (dropbox.sharing.RemoveFolderMemberArg attribute), 151
shared_folder_id (dropbox.sharing.SharedFolderMetadata attribute), 165
shared_folder_id (dropbox.sharing.TransferFolderArg attribute), 172
shared_folder_id (dropbox.sharing.UnmountFolderArg attribute), 173
shared_folder_id (dropbox.sharing.UnshareFolderArg attribute), 175
shared_folder_id (dropbox.sharing.UpdateFolderMemberArg attribute), 177
shared_folder_id (dropbox.sharing.UpdateFolderPolicyArg attribute), 178
shared_folder_only (dropbox.sharing.ResolvedVisibility attribute), 154
shared_folder_only (dropbox.sharing.Visibility attribute), 181
shared_folders (dropbox.team.GetStorageReport attribute), 192
shared_link_access_denied (dropbox.sharing.SharedLinkError attribute), 167
shared_link_already_exists (dropbox.sharing.CreateSharedLinkWithSettingsError attribute), 108
shared_link_is_directory (dropbox.sharing.GetSharedLinkFileError attribute), 121
shared_link_malformed (dropbox.sharing.RevokeSharedLinkError attribute), 154
shared_link_not_found (dropbox.sharing.SharedLinkError attribute), 167
shared_link_policy (dropbox.sharing.FolderPolicy attribute), 117
shared_link_policy (dropbox.sharing.ShareFolderArg attribute), 155
shared_link_policy (dropbox.sharing.UpdateFolderPolicyArg attribute), 179
shared_links_created (dropbox.team.GetActivityReport attribute), 189
shared_links_viewed_by_not_logged_in (dropbox.team.GetActivityReport attribute), 190
shared_links_viewed_by_outside_user (dropbox.team.GetActivityReport attribute), 190
shared_links_viewed_by_team (dropbox.team.GetActivityReport attribute), 190
shared_links_viewed_total (dropbox.team.GetActivityReport attribute), 190
shared_usage (dropbox.team.GetStorageReport attribute), 192
SharedFileMembers (class in dropbox.sharing), 160
SharedFileMetadata (class in dropbox.sharing), 160
SharedFolderAccessError (class in dropbox.sharing), 162
SharedFolderMemberError (class in dropbox.sharing), 163
SharedFolderMembers (class in dropbox.sharing), 163
SharedFolderMetadata (class in dropbox.sharing), 164
SharedFolderMetadataBase (class in dropbox.sharing), 165
SharedLinkAccessFailureReason (class in dropbox.sharing), 166
SharedLinkError (class in dropbox.sharing), 167
SharedLinkMetadata (class in dropbox.sharing), 167
SharedLinkPolicy (class in dropbox.sharing), 168
SharedLinkSettings (class in dropbox.sharing), 169
SharedLinkSettingsError (class in dropbox.sharing), 169
ShareFolderArg (class in dropbox.sharing), 154
ShareFolderError (class in dropbox.sharing), 155
ShareFolderErrorBase (class in dropbox.sharing), 155
ShareFolderJobStatus (class in dropbox.sharing), 156
ShareFolderLaunch (class in dropbox.sharing), 157
SharePathError (class in dropbox.sharing), 157
sharing_add_file_member() (dropbox.dropbox.Dropbox method), 20
sharing_add_folder_member() (dropbox.dropbox.Dropbox method), 20
sharing_change_file_member_access() (dropbox.dropbox.Dropbox method), 21
sharing_check_job_status() (dropbox.dropbox.Dropbox method), 21
sharing_check_remove_member_job_status() (dropbox.dropbox.Dropbox method), 21
sharing_check_share_job_status() (dropbox.dropbox.Dropbox method), 21
sharing_create_shared_link() (dropbox.dropbox.Dropbox method), 22
sharing_create_shared_link_with_settings() (dropbox.dropbox.Dropbox method), 22
sharing_get_file_metadata() (dropbox.dropbox.Dropbox method), 22
sharing_get_file_metadata_batch() (dropbox.dropbox.Dropbox method), 23
sharing_get_folder_metadata() (dropbox.dropbox.Dropbox method), 23

sharing_get_shared_link_file() (dropbox.dropbox.Dropbox method), 23
sharing_get_shared_link_file_to_file() (dropbox.dropbox.Dropbox method), 23
sharing_get_shared_link_metadata() (dropbox.dropbox.Dropbox method), 24
sharing_get_shared_links() (dropbox.dropbox.Dropbox method), 24
sharing_info (dropbox.files.FileMetadata attribute), 56
sharing_info (dropbox.files.FolderMetadata attribute), 57
sharing_list_file_members() (dropbox.dropbox.Dropbox method), 24
sharing_list_file_members_batch() (dropbox.dropbox.Dropbox method), 25
sharing_list_file_members_continue() (dropbox.dropbox.Dropbox method), 25
sharing_list_folder_members() (dropbox.dropbox.Dropbox method), 25
sharing_list_folder_members_continue() (dropbox.dropbox.Dropbox method), 25
sharing_list_folders() (dropbox.dropbox.Dropbox method), 26
sharing_list_folders_continue() (dropbox.dropbox.Dropbox method), 26
sharing_list_mountable_folders() (dropbox.dropbox.Dropbox method), 26
sharing_list_mountable_folders_continue() (dropbox.dropbox.Dropbox method), 26
sharing_list_received_files() (dropbox.dropbox.Dropbox method), 27
sharing_list_received_files_continue() (dropbox.dropbox.Dropbox method), 27
sharing_list_shared_links() (dropbox.dropbox.Dropbox method), 27
sharing_modify_shared_link_settings() (dropbox.dropbox.Dropbox method), 27
sharing_mount_folder() (dropbox.dropbox.Dropbox method), 28
sharing_policies (dropbox.users.FullTeam attribute), 248
sharing_relinquish_file_membership() (dropbox.dropbox.Dropbox method), 28
sharing_relinquish_folder_membership() (dropbox.dropbox.Dropbox method), 28
sharing_remove_file_member() (dropbox.dropbox.Dropbox method), 28
sharing_remove_file_member_2() (dropbox.dropbox.Dropbox method), 29
sharing_remove_folder_member() (dropbox.dropbox.Dropbox method), 29
sharing_revoke_shared_link() (dropbox.dropbox.Dropbox method), 29
sharing_share_folder() (dropbox.dropbox.Dropbox method), 30
sharing_transfer_folder() (dropbox.dropbox.Dropbox method), 30
sharing_unmount_folder() (dropbox.dropbox.Dropbox method), 30
sharing_unshare_file() (dropbox.dropbox.Dropbox method), 31
sharing_unshare_folder() (dropbox.dropbox.Dropbox method), 31
sharing_update_folder_member() (dropbox.dropbox.Dropbox method), 31
sharing_update_folder_policy() (dropbox.dropbox.Dropbox method), 31
SharingFileAccessError (class in dropbox.sharing), 170
SharingInfo (class in dropbox.files), 84
SharingUserError (class in dropbox.sharing), 171
short_url (dropbox.sharing.CreateSharedLinkArg attribute), 106
size (dropbox.files.FileMetadata attribute), 56
size (dropbox.files.ThumbnailArg attribute), 85
size (dropbox.sharing.FileLinkMetadata attribute), 111
space_left (dropbox.sharing.InsufficientQuotaAmounts attribute), 123
space_needed (dropbox.sharing.InsufficientQuotaAmounts attribute), 123
space_shortage (dropbox.sharing.InsufficientQuotaAmounts attribute), 123
SpaceAllocation (class in dropbox.users), 250
SpaceUsage (class in dropbox.users), 251
start (dropbox.files.SearchArg attribute), 82
start (dropbox.files.SearchResult attribute), 84
start() (dropbox.oauth.DropboxOAuth2Flow method), 4
start() (dropbox.oauth.DropboxOAuth2FlowNoRedirect method), 5
start_date (dropbox.team.BaseDfbReport attribute), 184
start_date (dropbox.team.DateRange attribute), 184
status (dropbox.team.MemberProfile attribute), 220
StorageBucket (class in dropbox.team), 239
success (dropbox.team.RevokeDeviceSessionStatus attribute), 237
success (dropbox.team.RevokeLinkedAppStatus attribute), 239
success() (dropbox.files.DeleteBatchResultEntry class method), 53
success() (dropbox.files.UploadSessionFinishBatchResultEntry class method), 91
success() (dropbox.sharing.FileMemberActionIndividualResult class method), 112
success() (dropbox.sharing.FileMemberRemoveActionResult class method), 114
success() (dropbox.team.MemberAddResult class method), 217
support_admin (dropbox.team.AdminTier attribute), 183
surname (dropbox.users.Name attribute), 250

suspend_inactive_user (dropbox.team.MembersSuspendError attribute), 232
suspend_last_admin (dropbox.team.MembersSuspendError attribute), 232
suspended (dropbox.team.TeamMemberStatus attribute), 241
suspended_members (dropbox.team.GetMembershipReport attribute), 191

T

target_is_indirect_member (dropbox.sharing.PermissionDeniedReason attribute), 146
target_is_owner (dropbox.sharing.PermissionDeniedReason attribute), 146
target_is_self (dropbox.sharing.PermissionDeniedReason attribute), 146
target_not_active (dropbox.sharing.PermissionDeniedReason attribute), 146
Team (class in dropbox.users), 251
team (dropbox.sharing.MemberPolicy attribute), 140
team (dropbox.users.FullAccount attribute), 248
team() (dropbox.users.SpaceAllocation class method), 251
team_admin (dropbox.team.AdminTier attribute), 183
team_alpha_groups_create() (dropbox.dropbox.DropboxTeam method), 34
team_alpha_groups_get_info() (dropbox.dropbox.DropboxTeam method), 34
team_alpha_groups_list() (dropbox.dropbox.DropboxTeam method), 34
team_alpha_groups_list_continue() (dropbox.dropbox.DropboxTeam method), 34
team_alpha_groups_update() (dropbox.dropbox.DropboxTeam method), 35
team_and_password (dropbox.sharing.ResolvedVisibility attribute), 154
team_and_password (dropbox.sharing.Visibility attribute), 181
team_devices_list_member_devices() (dropbox.dropbox.DropboxTeam method), 35
team_devices_list_members_devices() (dropbox.dropbox.DropboxTeam method), 35
team_devices_list_team_devices() (dropbox.dropbox.DropboxTeam method), 36
team_devices_revoke_device_session() (dropbox.dropbox.DropboxTeam method), 36
team_devices_revoke_device_session_batch() (dropbox.dropbox.DropboxTeam method), 36
team_folder (dropbox.sharing.AddFolderMemberError attribute), 103
team_folder (dropbox.sharing.RelinquishFolderMembershipError attribute), 149
team_folder (dropbox.sharing.RemoveFolderMemberError attribute), 152
team_folder (dropbox.sharing.TransferFolderError attribute), 173
team_folder (dropbox.sharing.UnshareFolderError attribute), 176
team_get_info() (dropbox.dropbox.DropboxTeam method), 36
team_groups_create() (dropbox.dropbox.DropboxTeam method), 36
team_groups_delete() (dropbox.dropbox.DropboxTeam method), 37
team_groups_get_info() (dropbox.dropbox.DropboxTeam method), 37
team_groups_job_status_get() (dropbox.dropbox.DropboxTeam method), 37
team_groups_list() (dropbox.dropbox.DropboxTeam method), 37
team_groups_list_continue() (dropbox.dropbox.DropboxTeam method), 38
team_groups_members_add() (dropbox.dropbox.DropboxTeam method), 38
team_groups_members_list() (dropbox.dropbox.DropboxTeam method), 38
team_groups_members_list_continue() (dropbox.dropbox.DropboxTeam method), 38
team_groups_members_remove() (dropbox.dropbox.DropboxTeam method), 38
team_groups_members_set_access_type() (dropbox.dropbox.DropboxTeam method), 39
team_groups_update() (dropbox.dropbox.DropboxTeam method), 39
team_id (dropbox.team.TeamGetInfoResult attribute), 240
team_info (dropbox.sharing.TeamMemberInfo attribute), 172
team_license_limit (dropbox.team.MembersRecoverError attribute), 226
team_license_limit (dropbox.team.MembersSetPermissionsError attribute), 230
team_license_limit (dropbox.team.MembersSuspendError attribute), 232
team_license_limit (dropbox.team.MembersUnsuspendError attribute), 233
team_license_limit() (dropbox.team.MemberAddResult class method), 217

team_linked_apps_list_member_linked_apps() (dropbox.dropbox.DropboxTeam method), 39
team_linked_apps_list_members_linked_apps() (dropbox.dropbox.DropboxTeam method), 40
team_linked_apps_list_team_linked_apps() (dropbox.dropbox.DropboxTeam method), 40
team_linked_apps_revoke_linked_app() (dropbox.dropbox.DropboxTeam method), 40
team_linked_apps_revoke_linked_app_batch() (dropbox.dropbox.DropboxTeam method), 40
team_member_id (dropbox.sharing.UserInfo attribute), 180
team_member_id (dropbox.team.DeviceSessionArg attribute), 187
team_member_id (dropbox.team.ListMemberAppsArg attribute), 206
team_member_id (dropbox.team.ListMemberDevicesArg attribute), 207
team_member_id (dropbox.team.MemberDevices attribute), 218
team_member_id (dropbox.team.MemberLinkedApps attribute), 219
team_member_id (dropbox.team.MemberProfile attribute), 220
team_member_id (dropbox.team.MembersSetPermissionsResult attribute), 230
team_member_id (dropbox.team.RevokeLinkedApiAppArg attribute), 237
team_member_id (dropbox.users.BasicAccount attribute), 247
team_member_id (dropbox.users.FullAccount attribute), 248
team_member_id() (dropbox.team.UserSelectorArg class method), 243
team_member_ids() (dropbox.team.UsersSelectorArg class method), 245
team_member_info (dropbox.sharing.SharedLinkMetadata attribute), 168
team_members_add() (dropbox.dropbox.DropboxTeam method), 41
team_members_add_job_status_get() (dropbox.dropbox.DropboxTeam method), 41
team_members_get_info() (dropbox.dropbox.DropboxTeam method), 41
team_members_list() (dropbox.dropbox.DropboxTeam method), 41
team_members_list_continue() (dropbox.dropbox.DropboxTeam method), 42
team_members_recover() (dropbox.dropbox.DropboxTeam method), 42
team_members_remove() (dropbox.dropbox.DropboxTeam method), 42
team_members_remove_job_status_get() (dropbox.dropbox.DropboxTeam method), 42
team_members_send_welcome_email() (dropbox.dropbox.DropboxTeam method), 43
team_members_set_admin_permissions() (dropbox.dropbox.DropboxTeam method), 43
team_members_set_profile() (dropbox.dropbox.DropboxTeam method), 43
team_members_suspend() (dropbox.dropbox.DropboxTeam method), 44
team_members_unsuspend() (dropbox.dropbox.DropboxTeam method), 44
team_only (dropbox.sharing.RequestedVisibility attribute), 153
team_only (dropbox.sharing.SharedLinkAccessFailureReason attribute), 167
team_only (dropbox.sharing.Visibility attribute), 181
team_policy_disallows_member_policy (dropbox.sharing.ShareFolderErrorBase attribute), 156
team_policy_disallows_member_policy (dropbox.sharing.UpdateFolderPolicyError attribute), 180
team_properties_template_add() (dropbox.dropbox.DropboxTeam method), 44
team_properties_template_get() (dropbox.dropbox.DropboxTeam method), 44
team_properties_template_list() (dropbox.dropbox.DropboxTeam method), 44
team_properties_template_update() (dropbox.dropbox.DropboxTeam method), 44
team_reports_get_activity() (dropbox.dropbox.DropboxTeam method), 45
team_reports_get_devices() (dropbox.dropbox.DropboxTeam method), 45
team_reports_get_membership() (dropbox.dropbox.DropboxTeam method), 45
team_reports_get_storage() (dropbox.dropbox.DropboxTeam method), 46
team_size (dropbox.team.GetMembershipReport attribute), 191
TeamGetInfoResult (class in dropbox.team), 239
TeamMemberInfo (class in dropbox.sharing), 171
TeamMemberInfo (class in dropbox.team), 240
TeamMemberProfile (class in dropbox.team), 240
TeamMembershipType (class in dropbox.team), 241
TeamMemberStatus (class in dropbox.team), 240
TeamSpaceAllocation (class in dropbox.users), 252
template_id (dropbox.files.PropertyGroupUpdate attribute), 71
template_id (dropbox.team.AddPropertyTemplateResult attribute), 182

template_id (dropbox.team.UpdatePropertyTemplateArg attribute), 242
template_id (dropbox.team.UpdatePropertyTemplateResult attribute), 242
ThumbnailArg (class in dropbox.files), 85
ThumbnailError (class in dropbox.files), 85
ThumbnailFormat (class in dropbox.files), 86
ThumbnailSize (class in dropbox.files), 86
time_invited (dropbox.sharing.SharedFileMetadata attribute), 162
time_invited (dropbox.sharing.SharedFolderMetadata attribute), 165
time_taken (dropbox.files.MediaMetadata attribute), 69
timeout (dropbox.files.ListFolderLongpollArg attribute), 64
to() (dropbox.files.RelocationError class method), 75
to_dropbox_id (dropbox.sharing.TransferFolderArg attribute), 172
to_path (dropbox.files.RelocationPath attribute), 76
too_many_files (dropbox.files.RelocationError attribute), 76
too_many_files (dropbox.files.SaveCopyReferenceError attribute), 79
too_many_invitees (dropbox.sharing.AddFolderMemberError attribute), 103
too_many_members() (dropbox.sharing.AddFolderMemberError class method), 103
too_many_pending_invites() (dropbox.sharing.AddFolderMemberError class method), 103
too_many_shared_folder_targets (dropbox.files.UploadSessionFinishError attribute), 92
too_many_write_operations (dropbox.files.DeleteBatchError attribute), 51
too_many_write_operations (dropbox.files.RelocationBatchError attribute), 73
total (dropbox.team.DevicesActive attribute), 188
total_usage (dropbox.team.GetStorageReport attribute), 192
transfer_admin_id (dropbox.team.MembersRemoveArg attribute), 226
transfer_admin_is_not_admin (dropbox.team.MembersRemoveError attribute), 228
transfer_admin_user_not_found (dropbox.team.MembersRemoveError attribute), 228
transfer_admin_user_not_in_team (dropbox.team.MembersRemoveError attribute), 228
transfer_dest_id (dropbox.team.MembersRemoveArg attribute), 226
transfer_dest_user_not_found (dropbox.team.MembersRemoveError attribute), 228
transfer_dest_user_not_in_team (dropbox.team.MembersRemoveError attribute), 228
TransferFolderArg (class in dropbox.sharing), 172
TransferFolderError (class in dropbox.sharing), 172
traverse_only (dropbox.files.FolderSharingInfo attribute), 58

U

unmount (dropbox.sharing.FolderAction attribute), 116
unmounted (dropbox.sharing.SharedFolderAccessError attribute), 163
UnmountFolderArg (class in dropbox.sharing), 173
UnmountFolderError (class in dropbox.sharing), 173
unshare (dropbox.sharing.FileAction attribute), 109
unshare (dropbox.sharing.FolderAction attribute), 116
unshare_folder_error() (dropbox.sharing.JobError class method), 125
unshared_usage (dropbox.team.GetStorageReport attribute), 192
UnshareFileArg (class in dropbox.sharing), 174
UnshareFileError (class in dropbox.sharing), 174
UnshareFolderArg (class in dropbox.sharing), 175
UnshareFolderError (class in dropbox.sharing), 175
unspecified_transfer_admin_id (dropbox.team.MembersRemoveError attribute), 228
unsupported_content (dropbox.files.PreviewError attribute), 71
unsupported_extension (dropbox.files.PreviewError attribute), 71
unsupported_extension (dropbox.files.ThumbnailError attribute), 86
unsupported_image (dropbox.files.ThumbnailError attribute), 86
unsuspend_non_suspended_member (dropbox.team.MembersUnsuspendError attribute), 233
unverified_dropbox_id() (dropbox.sharing.AddMemberSelectorError class method), 105
update() (dropbox.files.WriteMode class method), 97
update_property_groups (dropbox.files.UpdatePropertyGroupArg attribute), 88
updated (dropbox.team.DeviceSession attribute), 186
UpdateFolderMemberArg (class in dropbox.sharing), 176
UpdateFolderMemberError (class in dropbox.sharing), 177

UpdateFolderPolicyArg (class in dropbox.sharing), 178
UpdateFolderPolicyError (class in dropbox.sharing), 179
UpdatePropertiesError (class in dropbox.files), 87
UpdatePropertyGroupArg (class in dropbox.files), 87
UpdatePropertyTemplateArg (class in dropbox.team), 242
UpdatePropertyTemplateResult (class in dropbox.team), 242
upload_session_id (dropbox.files.UploadWriteFailed attribute), 94
UploadError (class in dropbox.files), 88
UploadErrorWithProperties (class in dropbox.files), 88
UploadSessionAppendArg (class in dropbox.files), 88
UploadSessionCursor (class in dropbox.files), 89
UploadSessionFinishArg (class in dropbox.files), 89
UploadSessionFinishBatchArg (class in dropbox.files), 89
UploadSessionFinishBatchJobStatus (class in dropbox.files), 90
UploadSessionFinishBatchResult (class in dropbox.files), 90
UploadSessionFinishBatchResultEntry (class in dropbox.files), 90
UploadSessionFinishError (class in dropbox.files), 91
UploadSessionLookupError (class in dropbox.files), 92
UploadSessionOffsetError (class in dropbox.files), 93
UploadSessionStartArg (class in dropbox.files), 93
UploadSessionStartResult (class in dropbox.files), 93
UploadWriteFailed (class in dropbox.files), 93
url (dropbox.files.SaveUrlArg attribute), 80
url (dropbox.sharing.GetSharedLinkMetadataArg attribute), 121
url (dropbox.sharing.LinkMetadata attribute), 126
url (dropbox.sharing.ModifySharedLinkSettingsArgs attribute), 142
url (dropbox.sharing.RevokeSharedLinkArg attribute), 154
url (dropbox.sharing.SharedLinkMetadata attribute), 168
used (dropbox.users.SpaceUsage attribute), 251
used (dropbox.users.TeamSpaceAllocation attribute), 252
user (dropbox.sharing.InviteeMembershipInfo attribute), 124
user (dropbox.sharing.UserMembershipInfo attribute), 180
user (dropbox.team.GroupMemberSelector attribute), 195
user (dropbox.team.MemberAccess attribute), 214
user (dropbox.team.MembersDeactivateArg attribute), 222
user (dropbox.team.MembersRecoverArg attribute), 225
user (dropbox.team.MembersSetPermissionsArg attribute), 229
user (dropbox.team.MembersSetProfileArg attribute), 231
user (dropbox.team.MembersUnsuspendArg attribute), 232
user_agent (dropbox.team.ActiveWebSession attribute), 182
user_already_on_team() (dropbox.team.MemberAddResult class method), 217
user_already_paired() (dropbox.team.MemberAddResult class method), 217
user_cannot_be_manager_of_company_managed_group (dropbox.team.GroupMemberSetAccessTypeError attribute), 195
user_cannot_be_manager_of_company_managed_group() (dropbox.team.GroupMembersAddError class method), 197
user_creation_failed() (dropbox.team.MemberAddResult class method), 218
user_error() (dropbox.sharing.AddFileMemberError class method), 100
user_error() (dropbox.sharing.GetFileMetadataError class method), 119
user_error() (dropbox.sharing.ListFileMembersContinueError class method), 129
user_error() (dropbox.sharing.ListFileMembersError class method), 130
user_error() (dropbox.sharing.ListFilesContinueError class method), 132
user_error() (dropbox.sharing.RemoveFileMemberError class method), 150
user_error() (dropbox.sharing.UnshareFileError class method), 175
user_management_admin (dropbox.team.AdminTier attribute), 183
user_migration_failed() (dropbox.team.MemberAddResult class method), 218
user_must_be_active_to_be_owner (dropbox.team.GroupMembersAddError attribute), 197
user_not_allowed_by_owner (dropbox.sharing.PermissionDeniedReason attribute), 146
user_not_found (dropbox.team.UserSelectorError attribute), 244
user_not_in_team (dropbox.team.MembersDeactivateError attribute), 222
user_not_in_team (dropbox.team.MemberSelectorError attribute), 220
user_not_in_team (dropbox.team.MembersRecoverError attribute), 226
user_not_in_team (dropbox.team.MembersSetPermissionsError attribute), 230

user_not_same_team_as_owner (dropbox.sharing.PermissionDeniedReason attribute), 146
user_on_another_team() (dropbox.team.MemberAddResult class method), 218
user_unrecoverable (dropbox.team.MembersRecoverError attribute), 226
UserInfo (class in dropbox.sharing), 180
UserMembershipInfo (class in dropbox.sharing), 180
users (dropbox.sharing.SharedFileMembers attribute), 160
users (dropbox.sharing.SharedFolderMembers attribute), 164
users (dropbox.team.GroupMembersRemoveArg attribute), 198
users (dropbox.team.GroupMembersSelector attribute), 198
users (dropbox.team.StorageBucket attribute), 239
users_get_account() (dropbox.dropbox.Dropbox method), 32
users_get_account_batch() (dropbox.dropbox.Dropbox method), 32
users_get_current_account() (dropbox.dropbox.Dropbox method), 32
users_get_space_usage() (dropbox.dropbox.Dropbox method), 32
users_not_found() (dropbox.team.GroupMembersAddError class method), 197
UserSelectorArg (class in dropbox.team), 243
UserSelectorError (class in dropbox.team), 243
UsersSelectorArg (class in dropbox.team), 244

V

VideoMetadata (class in dropbox.files), 94
viewer (dropbox.sharing.AccessLevel attribute), 98
viewer_no_comment (dropbox.sharing.AccessLevel attribute), 98
Visibility (class in dropbox.sharing), 180
visibility (dropbox.sharing.LinkMetadata attribute), 126

W

w1024h768 (dropbox.files.ThumbnailSize attribute), 87
w128h128 (dropbox.files.ThumbnailSize attribute), 87
w32h32 (dropbox.files.ThumbnailSize attribute), 87
w640h480 (dropbox.files.ThumbnailSize attribute), 87
w64h64 (dropbox.files.ThumbnailSize attribute), 87
warning (dropbox.sharing.MemberAccessLevelResult attribute), 138
web_session() (dropbox.team.RevokeDeviceSessionArg class method), 236
web_sessions (dropbox.team.MemberDevices attribute), 218
width (dropbox.files.Dimensions attribute), 54
windows (dropbox.team.DesktopPlatform attribute), 186
windows (dropbox.team.DevicesActive attribute), 188
windows_phone (dropbox.team.MobileClientPlatform attribute), 234
wipe_data (dropbox.team.MembersDeactivateArg attribute), 222
WriteConflictError (class in dropbox.files), 94
WriteError (class in dropbox.files), 94
WriteMode (class in dropbox.files), 96