DotNetBox Documentation

Release 1.1.1

nicoco007

January 18, 2016

Contents

CHAPTER 1

The .NET Framework for Dropbox

1.1 Getting Started

1.1.1 Download

DotNetBox can be downloaded on NuGet or by using the NuGet Package Manager. If you are using Visual Studio, do the following:

- Open the package manager by going to *Tools* > *NuGet Package Manager* > *Package Manager Console*
- Type `Install-Package DotNetBox`.

1.1.2 Using the DropboxClient Class

There are two ways to create an instance of the `DropboxClient` class. You can either create it using an access token you have already retrieved,

```
DropboxClient client = new DropboxClient("YOUR_ACCESS_TOKEN");
```

or by using your registered app's key and secret (get them from the My Apps section of the Dropbox Developers website).

```
DropboxClient client = new DropboxClient("APP_KEY", "APP_SECRET");
```

Using an access token will instantly give you the ability to call any API endpoint. However, if you chose the latter, you will need to authorize the user by redirecting them to the authorization webpage. To get the URL of this page, you need to call the `GetAuthorizeUrl()` function. You will need to specify a response type to call the function. For the sake of this tutorial, we will be using the Code flow.

```
string authorizeUrl = client.GetAuthorizeUrl(ResponseType.Code);
```

Using the Code flow means that the authorization web page will display, once the user accepts, a code that the user then has to enter into your app. You will need to pass this code to the `AuthorizeCode()` function.

```
await client.AuthorizeCode("USER'S_CODE");
```

Here is an example of a simple authorization client:

```
DropboxClient client = new DropboxClient("APP_KEY", "APP_SECRET"); // create the client

private void GetCodeButton_Click()
{
```

```
        Process.Start(client.GetAuthorizeUrl(ResponseType.Code)) // this will open the authorization
}

private async void GetAccessTokenButton_Click()
{
        await client.AuthorizeCode(CodeTextBox.Text); // authorize the code the user entered in a te
}
```

You will now be able to call API endpoints.

1.2 Calling your first endpoint

1.2.1 Getting the user's information

You are now ready to use Dropbox's API! To start off, get the current user's information by calling the GetCurrentAccount() function.

```
FullAccount currentAccount = await client.Users.GetCurrentAccount();
```

This will give you access to information such as the user's account ID, their name and email, their referral link, their account type, and more! You can use this information to display who is connected to your application.

You can also get the current user's space usage by calling the GetSpaceUsage() function.

```
SpaceUsage spaceUsage = await client.Users.GetSpaceUsage();
```

This gives you the user's current space usage and their space quota.

1.2.2 File management

Dropbox is a file storage service; therefore, there must be API endpoints for file management, right? Of course! We will cover the most basic ones, but you can check out all of them in the left sidebar, and most of them are quite self-explanatory.

Getting metadata

The first endpoint we will use is the GetMetadata() function. It retrieves a file or folder's metadata. It is important to either call FileExists() or FolderExists() before calling this function, because it will throw an exception if the file or folder doesn't exist.

```
if (client.Files.FileExists("/some_file.txt")) // check if the file exists
{
        FileMetadata fileMetadata = await client.Files.GetMetadata("/some_file.txt"); // get the file
}

if (client.Files.FolderExists("/some_folder")) // check if the folder exists
{
        FolderMetadata folderMetadata = await client.Files.GetMetadata("/some_folder"); // get the fo
}
```

It is important to note that all paths should start with a forwardslash (/) except if you are referencing the root directory, where you leave the path blank.

File operations

Copying and moving files

Coming soon!

1.3 Users.GetAccount Method (string)

Get information about a user's account.

1.3.1 Syntax

```
Users.GetAccount(string accountId)
```

1.3.2 Parameters

accountId *System.String* A user's account identifier.

1.3.3 Returns

`DotNetBox.BasicAccount` Basic information about the specified account.

1.4 Users.GetAccountBatch Method (string[])

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

1.4.1 Syntax

```
Users.GetAccountBatch(string[] accountIds)
```

1.4.2 Parameters

accountIds *System.String[]* List of user account identifiers. Should not contain any duplicate account IDs.

1.4.3 Returns

`DotNetBox.BasicAccount[]` List of basic information about every account.

1.5 Users.GetCurrentAccount Method

Get information about the current user's account.

1.5.1 Syntax

```
Users.GetCurrentAccount()
```

1.5.2 Returns

`DotNetBox.FullAccount` Detailed information about the current user's account.

1.6 Users.GetSpaceUsage Method

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

1.6.1 Syntax

```
Users.GetSpaceUsage()
```

1.6.2 Returns

`DotNetBox.SpaceUsage` Information about the user's space usage and quota.

1.7 Files.Cancel Method

Cancel the current asynchronous operation.

1.7.1 Syntax

```
Files.Cancel()
```

1.8 Files.Copy Method (string, string)

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.

1.8.1 Syntax

```
Files.Copy(string fromPath, string toPath)
```

1.8.2 Parameters

fromPath *System.String* Path in the user's Dropbox to be copied or moved.

toPath *System.String* Path in the user's Dropbox that is the destination.

1.8.3 Returns

`DotNetBox.Metadata` Metadata of the copied (destination) file.

1.9 Files.CreateFolder Method (string)

Create a folder at a given path.

1.9.1 Syntax

```
Files.CreateFolder(string path)
```

1.9.2 Parameters

path *System.String* Path in the user's Dropbox to create.

1.9.3 Returns

`DotNetBox.FolderMetadata` Metadata of the newly created folder.

1.10 Files.Delete Method (string)

Delete the file or folder at a given path. If the path is a folder, all its contents will be deleted too.

1.10.1 Syntax

```
Files.Delete(string path)
```

1.10.2 Parameters

path *System.String* Path in the user's Dropbox to delete.

1.10.3 Returns

`DotNetBox.Metadata` Metadata of the deleted file or folder.

1.11 Files.Download Method (string, string)

Download a file from a user's

1.11.1 Syntax

```
Files.Download(string path, string savePath)
```

1.11.2 Parameters

path *System.String* The path of the file to download.

savePath *System.String* The path to where the file will be saved.

1.11.3 Returns

`DotNetBox.FileMetadata` Metadata of the downloaded file.

1.12 Files.FileExists Method (string)

Checks whether a file exists or not.

1.12.1 Syntax

```
Files.FileExists(string path)
```

1.12.2 Parameters

path *System.String* Path of the file.

1.12.3 Returns

System.Boolean Whether the file exists or not.

1.13 Files.FolderExists Method (string)

Checks whether a folder exists or not.

1.13.1 Syntax

```
Files.FolderExists(string path)
```

1.13.2 Parameters

path *System.String* Path of the folder.

1.13.3 Returns

System.Boolean Whether the folder exists or not.

1.14 Files.GetMetadata Method (string, bool)

Returns the metadata for a file or folder.

1.14.1 Syntax

```
Files.GetMetadata(string path, bool includeMediaInfo)
```

1.14.2 Parameters

path *System.String* The path of a file or folder onCan also be a rev or id.

includeMediaInfo *System.Boolean* If true, FileMetadata.MediaInfo is set for photo and video. The default for this field is False.

1.14.3 Returns

`DotNetBox.Metadata` Metadata for a file or folder. It will be of type FileMetadata, FolderMetadata, or DeletedMetadata

1.15 Files.GetPreview Method (string, string)

Get a preview for a file in PDF format. 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

1.15.1 Syntax

```
Files.GetPreview(string path, string savePath)
```

1.15.2 Parameters

path *System.String* The path of the file to preview.

savePath *System.String* Path at which to save the preview file.

1.15.3 Returns

`DotNetBox.FileMetadata` Metadata of the file of which a preview was made.

1.16 Files.GetThumbnail Method (string, string, ThumbnailFormat, ThumbnailSize)

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.

1.16.1 Syntax

```
Files.GetThumbnail(string path, string savePath, ThumbnailFormat format, ThumbnailSize size)
```

1.16.2 Parameters

path *System.String* The path to the image file you want to thumbnail.

savePath *System.String* The path at which to save the thumbnail image.

format `DotNetBox.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 `DotNetBox.ThumbnailSize` The size for the thumbnail image.

1.16.3 Returns

`DotNetBox.FileMetadata` Metadata of the file of which a thumbnail was made.

1.17 Files.ListFolder Method (string, bool, bool, bool)

Returns the contents of a folder.

1.17.1 Syntax

```
Files.ListFolder(string path, bool recursive, bool includeMediaInfo, bool includeDeleted)
```

1.17.2 Parameters

path *System.String* The path to the folder you want to see the contents of.

recursive *System.Boolean* If true, the list folder operation will be applied recursively to all subfolders and the response will contain contents of all subfolders. The default for this field is False.

includeMediaInfo *System.Boolean* If true, FileMetadata.MediaInfo is set for photo and video. The default for this field is False.

includeDeleted *System.Boolean* If true, the results will include entries for files and folders that used to exist but were deleted. The default for this field is False.

1.17.3 Returns

`DotNetBox.ListFolderResult` Contents of folder.

1.18 Files.ListFolderContinue Method (string)

Once a cursor has been retrieved from ListFolder, use this to paginate through all files and retrieve updates to the folder.

1.18.1 Syntax

```
Files.ListFolderContinue(string cursor)
```

1.18.2 Parameters

cursor *System.String* The cursor returned by your last call to ListFolder or ListFolderContinue.

1.18.3 Returns

`DotNetBox.ListFolderResult` Contents of folder.

1.19 Files.ListFolderGetLatestCursor Method (string, bool, bool, bool)

A way to quickly get a cursor for the folder's state. Unlike ListFolder, ListFolderGetLatestCursor doesn't return any entries. This endpoint is for apps which only needs to know about new files and modifications and doesn't need to know about files that already exist in the folder.

1.19.1 Syntax

```
Files.ListFolderGetLatestCursor(string path, bool recursive, bool includeMediaInfo, bool includeDelet
```

1.19.2 Parameters

path *System.String* The path to the folder you want to see the contents of.

recursive *System.Boolean* If true, the list folder operation will be applied recursively to all subfolders and the response will contain contents of all subfolders. The default for this field is False.

includeMediaInfo *System.Boolean* If true, FileMetadata.MediaInfo is set for photo and video. The default for this field is False.

includeDeleted *System.Boolean* If true, the results will include entries for files and folders that used to exist but were deleted. The default for this field is False.

1.19.3 Returns

System.String Pass the cursor into ListFolderContinue to see what's changed in the folder since your previous query.

1.20 Files.ListFolderLongpoll Method (string, int)

A longpoll endpoint to wait for changes on an account. In conjunction with list_folder, 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.

1.20.1 Syntax

```
Files.ListFolderLongpoll(string cursor, int timeout)
```

1.20.2 Parameters

cursor *System.String* A cursor as returned by ListFolder or ListFolderContinue

timeout *System.Int32* 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. The default for this field is 30.

1.20.3 Returns

`DotNetBox.ListFolderLongpollResult` Whether files have been changed or not.

1.21 Files.ListRevisions Method (string, int)

Returns revisions of a file.

1.21.1 Syntax

```
Files.ListRevisions(string path, int limit)
```

1.21.2 Parameters

path *System.String* The path to the file you want to see the revisions of.

limit *System.Int32* The maximum number of revision entries returned. The default for this field is 10.

1.21.3 Returns

`DotNetBox.ListRevisionsResult` List of revisions applied to the file.

1.22 Files.Move Method (string, string)

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.

1.22.1 Syntax

```
Files.Move(string fromPath, string toPath)
```

1.22.2 Parameters

fromPath *System.String* Path in the user's Dropbox to be copied or moved.

toPath *System.String* Path in the user's Dropbox that is the destination.

1.22.3 Returns

`DotNetBox.Metadata` Metadata of the moved file.

1.23 Files.PermanentlyDelete Method (string)

Delete the file or folder at a given path. If the path is a folder, all its contents will be deleted too.

1.23.1 Syntax

```
Files.PermanentlyDelete(string path)
```

1.23.2 Parameters

path *System.String* Path in the user's Dropbox to delete.

1.23.3 Returns

`DotNetBox.Metadata` Metadata of the deleted file or folder.

1.24 Files.Restore Method (string, string)

Restore a file to a specific revision.

1.24.1 Syntax

```
Files.Restore(string path, string rev)
```

1.24.2 Parameters

path *System.String* The path to the file you want to restore.

rev *System.String* The revision to restore for the file. Defaults to previous revision.

1.24.3 Returns

`DotNetBox.FileMetadata` Metadata of the restored file.

1.25 Files.Search Method (string, string, int, int, SearchMode)

Searches for files and folders.

1.25.1 Syntax

```
Files.Search(string path, string query, int start, int maxResults, SearchMode mode)
```

1.25.2 Parameters

path *System.String* The path in the user's Dropbox to search. Should probably be a folder.

query *System.String* 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.

start *System.Int32* The starting index within the search results (used for paging). The default for this field is 0.

maxResults *System.Int32* The maximum number of search results to return. The default for this field is 100.

mode `DotNetBox.SearchMode` The search mode (Filename, FilenameAndContent, or DeletedFilename). Note that searching file content is only available for Dropbox Business accounts.

1.25.3 Returns

`DotNetBox.SearchResult` Files and folders that match the search query.

1.26 Files.Upload Method (string, string, WriteMode, bool, bool, string)

Create a new file with the contents provided in the request asynchronously.

1.26.1 Syntax

```
Files.Upload(string filePath, string path, WriteMode mode, bool autorename, bool mute, string rev)
```

1.26.2 Parameters

filePath *System.String* Path to the file to upload.

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

mode `DotNetBox.WriteMode` Selects what to do if the file already exists.

autorename *System.Boolean* If there's a conflict, as determined by mode, have the Dropbox server try to autorename the file to avoid conflict. The default for this field is False.

mute *System.Boolean* 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. The default for this field is False.

rev *System.String* 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. Only applies if WriteMode is Update.

1.27 Sharing.AddFolderMember Method (string, FolderMember[], bool, string)

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 mount_folder on their behalf. Apps must have full Dropbox access to use this endpoint. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.27.1 Syntax

```
Sharing.AddFolderMember(string sharedFolderId, FolderMember[] members, bool quiet, string customMessa
```

1.27.2 Parameters

sharedFolderId *System.String* The ID for the shared folder.

members `DotNetBox.FolderMember[]` The intended list of members to add. Added members will receive invites to join the shared folder.

quiet *System.Boolean* Whether added members should be notified via email and device notifications of their invite. The default for this field is False.

customMessage *System.String* Optional message to display to added members in their invitation. This field is optional.

1.28 Sharing.CheckJobStatus Method (string)

Returns the status of an asynchronous job. Apps must have full Dropbox access to use this endpoint. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.28.1 Syntax

```
Sharing.CheckJobStatus(string asyncJobId)
```

1.28.2 Parameters

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

1.28.3 Returns

`DotNetBox.JobStatus` Current status of the share job (in progress, completed, or failed).

1.29 Sharing.CheckShareJobStatus Method (string)

Returns the status of an asynchronous job for sharing a folder. Apps must have full Dropbox access to use this endpoint. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.29.1 Syntax

```
Sharing.CheckShareJobStatus(string asyncJobId)
```

1.29.2 Parameters

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

1.29.3 Returns

`DotNetBox.ShareFolderJobStatus` Current status of the share job (in progress, completed, or failed). If completed, returns `DotNetBox.SharedFolderMetadata` under `DotNetBox.ShareFolderJobStatus.FolderMetadata`.

1.30 Sharing.CreateSharedLink Method (string, RequestedVisibility?, string, DateTime?)

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).

1.30.1 Syntax

```
Sharing.CreateSharedLink(string path, RequestedVisibility? requestedVisibility, string linkPassword,
```

1.30.2 Parameters

path *System.String* The path to be shared by the shared link.

requestedVisibility *DotNetBox.RequestedVisibility?* The requested access for this shared link. This field is optional.

linkPassword *System.String* If RequestedVisibility is RequestedVisibility.Password this is needed to specify the password to access the link. This field is optional.

expires *System.DateTime?* Expiration time of the shared link. By default the link won't expire. This field is optional.

1.30.3 Returns

`DotNetBox.LinkMetadata` Metadata of the newly created link.

1.31 Sharing.DownloadSharedLinkFile Method (string, string, string, string)

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

1.31.1 Syntax

```
Sharing.DownloadSharedLinkFile(string savePath, string url, string path, string linkPassword)
```

1.31.2 Parameters

savePath *System.String* Path to which the file will be saved.

url *System.String* URL of the shared link.

path *System.String* 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. This field is optional.

linkPassword *System.String* If the shared link has a password, this parameter can be used. This field is optional.

1.31.3 Returns

`DotNetBox.LinkMetadata` Metadata of the shared link used to download the file.

1.32 Sharing.GetSharedFolder Method (string)

Retrieves a shared folder from its path.

1.32.1 Syntax

```
Sharing.GetSharedFolder(string path)
```

1.32.2 Parameters

path *System.String* Path of the folder to get.

1.32.3 Returns

`DotNetBox.SharedFolderMetadata`

1.33 Sharing.GetSharedFolderMetadata Method (string)

Returns shared folder metadata by its folder ID. Apps must have full Dropbox access to use this endpoint. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.33.1 Syntax

```
Sharing.GetSharedFolderMetadata(string sharedFolderId)
```

1.33.2 Parameters

sharedFolderId *System.String* The ID for the shared folder.

1.33.3 Returns

`DotNetBox.SharedFolderMetadata` Metadata of the shared folder (name, path, policies, etc.)

1.34 Sharing.GetSharedLink Method (string)

Gets a file or folder's shared link.

1.34.1 Syntax

```
Sharing.GetSharedLink(string path)
```

1.34.2 Parameters

path *System.String* Path of file/folder.

1.34.3 Returns

`DotNetBox.LinkMetadata` If the specified file/folder has a shared link, metadata of that link.

1.35 Sharing.GetSharedLinkMetadata Method (string, string, string)

Get the shared link's metadata.

1.35.1 Syntax

```
Sharing.GetSharedLinkMetadata(string url, string path, string password)
```

1.35.2 Parameters

url *System.String* URL of the shared link.

path *System.String* 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. This field is optional.

password *System.String* If the shared link has a password, this parameter can be used. This field is optional.

1.35.3 Returns

`DotNetBox.LinkMetadata` Metadata of the shared link.

1.36 Sharing.HasSharedLink Method (string)

Check if a file or folder has a shared link.

1.36.1 Syntax

```
Sharing.HasSharedLink(string path)
```

1.36.2 Parameters

path *System.String* Path of file/folder.

1.36.3 Returns

System.Boolean Whether the file or folder has a shared link or not.

1.37 Sharing.ListSharedFolderMembers Method (string)

Returns shared folder membership by its folder ID. Apps must have full Dropbox access to use this endpoint. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.37.1 Syntax

```
Sharing.ListSharedFolderMembers(string sharedFolderId)
```

1.37.2 Parameters

sharedFolderId *System.String* The ID for the shared folder

1.37.3 Returns

`DotNetBox.SharedFolderMembers` List of specified shared folder's members (users, groups, and invitees).

1.38 Sharing.ListSharedFolderMembersContinue Method (string)

Once a cursor has been retrieved from ListSharedFolderMembers, use this to paginate through all shared folder members. Apps must have full Dropbox access to use this endpoint. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.38.1 Syntax

```
Sharing.ListSharedFolderMembersContinue(string cursor)
```

1.38.2 Parameters

cursor *System.String* The cursor returned by your last call to ListSharedFolderMembers(string) or ListSharedFolderMembersContinue(string) .

1.38.3 Returns

`DotNetBox.SharedFolderMembers` Continued list of specified shared folder's members.

1.39 Sharing.ListSharedFolders Method

Return the list of all shared folders the current user has access to. Apps must have full Dropbox access to use this endpoint. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.39.1 Syntax

```
Sharing.ListSharedFolders()
```

1.39.2 Returns

`DotNetBox.ListSharedFoldersResult` List of user's shared folders and cursor if there are more folders. Pass cursor to Sharing.ListSharedFoldersContinue(string) to get additional folders.

1.40 Sharing.ListSharedFoldersContinue Method (string)

Once a cursor has been retrieved from ListSharedFolders() , use this to paginate through all shared folders. Apps must have full Dropbox access to use this endpoint. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.40.1 Syntax

```
Sharing.ListSharedFoldersContinue(string cursor)
```

1.40.2 Parameters

cursor *System.String*

1.40.3 Returns

`DotNetBox.ListSharedFoldersResult` Continued list of user's shared folders.

1.41 Sharing.ListSharedLinks Method (string, string)

List shared links of this user. If no path is given or the path is empty, 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.

1.41.1 Syntax

```
Sharing.ListSharedLinks(string path, string cursor)
```

1.41.2 Parameters

path *System.String* See summary.

cursor *System.String* The cursor returned by your last call to ListSharedLinks. This field is optional.

1.41.3 Returns

`DotNetBox.ListSharedLinksResult` List of the user's shared links.

1.42 Sharing.ModifySharedLinkSettings Method (string, RequestedVisibility?, string, DateTime?)

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 SharedLinkMetadata will reflect the actual visibility of the shared link and the LinkPermissions.requested_visibility will reflect the requested visibility.

1.42.1 Syntax

```
Sharing.ModifySharedLinkSettings(string url, RequestedVisibility? requestedVisibility, string linkPas
```

1.42.2 Parameters

url *System.String* URL of the shared link to change its settings.

requestedVisibility *DotNetBox.RequestedVisibility?* The requested access for this shared link. This field is optional.

linkPassword *System.String* If RequestedVisibility is RequestedVisibility.Password this is needed to specify the password to access the link. This field is optional.

expires *System.DateTime?* Expiration time of the shared link. By default the link won't expire. This field is optional.

1.42.3 Returns

`DotNetBox.LinkMetadata` Metadata of the modified shared link.

1.43 Sharing.MountFolder Method (string)

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. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.43.1 Syntax

```
Sharing.MountFolder(string sharedFolderId)
```

1.43.2 Parameters

sharedFolderId *System.String* The ID of the shared folder to mount.

1.43.3 Returns

`DotNetBox.SharedFolderMetadata`

1.44 Sharing.RelinquishFolderMembership Method (string)

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. Apps must have full Dropbox access to use this endpoint. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.44.1 Syntax

```
Sharing.RelinquishFolderMembership(string sharedFolderId)
```

1.44.2 Parameters

sharedFolderId *System.String* The ID for the shared folder.

1.45 Sharing.RemoveFolderMember Method (string, string, bool)

Allows an owner or editor(if the ACL update policy allows) of a shared folder to remove another member. Apps must have full Dropboxaccess to use this endpoint. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.45.1 Syntax

```
Sharing.RemoveFolderMember(string sharedFolderId, string memberIdentifier, bool leaveCopy)
```

1.45.2 Parameters

sharedFolderId *System.String* The ID for the shared folder.

memberIdentifier *System.String* The member to remove from the folder.

leaveCopy *System.Boolean* 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.

1.45.3 Returns

`DotNetBox.LaunchEmptyResult`

1.46 Sharing.RevokeSharedLink Method (string)

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 ListSharedLinks with the file as the ListSharedLinksArg.Path argument.

1.46.1 Syntax

```
Sharing.RevokeSharedLink(string url)
```

1.46.2 Parameters

url *System.String* URL of the shared link.

1.47 Sharing.ShareFolder Method (string, bool, AclUpdatePolicy, SharedLinkPolicy, MemberPolicy)

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 forceAsync. If a AsyncJobId is returned, you'll need to call CheckShareJobStatus until the action completes to get the metadata for the folder. Apps must have full Dropbox access to use this endpoint. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.47.1 Syntax

```
Sharing.ShareFolder(string path, bool forceAsync, AclUpdatePolicy aclUpdatePolicy, SharedLinkPolicy s
```

1.47.2 Parameters

path *System.String* The path to the folder to share. If it does not exist, then a new one is created.

forceAsync *System.Boolean* Whether to force the share to happen asynchronously. The default for this field is False.

aclUpdatePolicy `DotNetBox.AclUpdatePolicy` Who can add and remove members of this shared folder.

sharedLinkPolicy `DotNetBox.SharedLinkPolicy` The policy to apply to shared links created for content inside this shared folder.

memberPolicy `DotNetBox.MemberPolicy` Who can be a member of this shared folder.

1.47.3 Returns

`DotNetBox.ShareFolderLaunch` If asynchronous, returns async job ID under `DotNetBox.ShareFolderLaunch.AsyncJobId` . If not, returns `DotNetBox.SharedFolderMetadata` under `DotNetBox.ShareFolderLaunch.FolderMetadata` .

1.48 Sharing.TransferFolder Method (string, string)

Transfer ownership of a shared folder to a member of the shared folder. Apps must have full Dropbox access to use this endpoint. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.48.1 Syntax

```
Sharing.TransferFolder(string sharedFolderId, string dropboxId)
```

1.48.2 Parameters

sharedFolderId *System.String* The ID for the shared folder.

dropboxId *System.String* A account or team member ID to transfer ownership to.

1.49 Sharing.UnmountFolder Method (string)

Unmounts the designated folder. They can re-mount the folder at a later time using mount_folder. Apps must have full Dropbox access to use this endpoint. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.49.1 Syntax

```
Sharing.UnmountFolder(string sharedFolderId)
```

1.49.2 Parameters

sharedFolderId *System.String* The ID for the shared folder.

1.50 Sharing.UnshareFolder Method (string, bool)

Allows a shared folder owner to unshare the folder. You'll need to call CheckJobStatus(string) to determine if the action has completed successfully. Apps must have full Dropbox access to use this endpoint. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.50.1 Syntax

```
Sharing.UnshareFolder(string sharedFolderId, bool leaveCopy)
```

1.50.2 Parameters

sharedFolderId *System.String* The ID for the shared folder.

leaveCopy *System.Boolean* 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. Default value is False.

1.50.3 Returns

`DotNetBox.LaunchEmptyResult` Returns a job ID to be used with Sharing.CheckJobStatus(string) if in progress. If not, just returns `DotNetBox.AsyncJobStatus.Completed`.

1.51 Sharing.UpdateFolderMember Method (string, string, AccessLevel)

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. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.51.1 Syntax

```
Sharing.UpdateFolderMember(string sharedFolderId, string memberIdentifier, AccessLevel accessLevel)
```

1.51.2 Parameters

sharedFolderId *System.String* The ID for the shared folder.

memberIdentifier *System.String* The member of the shared folder to update.

accessLevel `DotNetBox.AccessLevel` The new access level for member. AccessLevel.owner is disallowed.

1.52 Sharing.UpdateFolderPolicy Method (string, AclUpdatePolicy?, SharedLinkPolicy?, MemberPolicy?)

Update the sharing policies for a shared folder. Apps must have full Dropbox access to use this endpoint. Warning: This endpoint is in beta and is subject to minor but possibly backwards-incompatible changes.

1.52.1 Syntax

```
Sharing.UpdateFolderPolicy(string sharedFolderId, AclUpdatePolicy? aclUpdatePolicy, SharedLinkPolicy
```

1.52.2 Parameters

sharedFolderId *System.String* The ID for the shared folder.

aclUpdatePolicy *DotNetBox.AclUpdatePolicy?* Who can add and remove members of this shared folder. This field is optional.

sharedLinkPolicy *DotNetBox.SharedLinkPolicy?* The policy to apply to shared links created for content inside this shared folder. This field is optional.

memberPolicy *DotNetBox.MemberPolicy?* Who can be a member of this shared folder. Only set this if the current user is on a team. This field is optional.

1.52.3 Returns

`DotNetBox.SharedFolderMetadata`

F

S

U