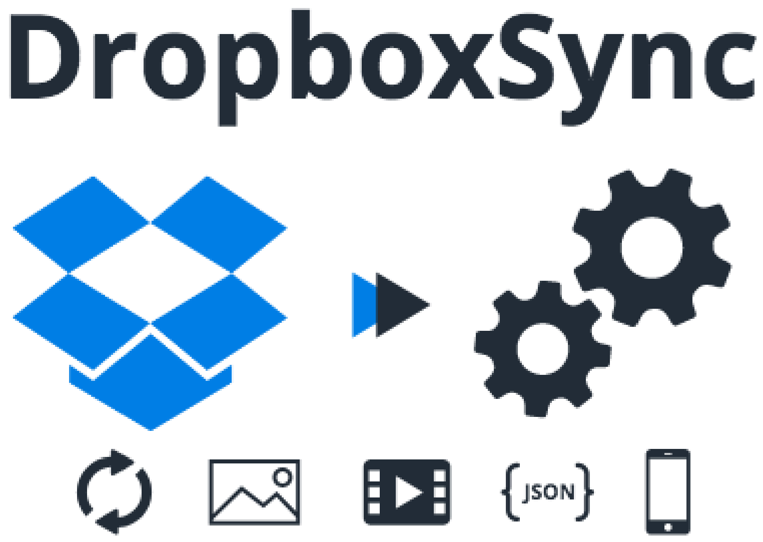
George Fedoseev

DropboxSync v2.0 `Documentation`



Contents

`Retrieving files`

GetFileAsBytes()

```
public void GetFileAsBytes(string dropboxPath,
                           Action<DropboxRequestResult<byte[]>> onResult,
                           Action<float> onProgress = null,
                           bool useCachedFirst = false,
                           bool useCachedIfOffline = true,
                           bool receiveUpdates = false)
```

Asynchronously retrieves file from Dropbox as `byte[]`.

Parameters

Parameter	Description
dropboxPath (`string`)	Path to file on Dropbox or inside of Dropbox App folder (depending on accessToken type). Should start with "/". *Example: /DropboxSyncExampleFolder/image.jpg*
onResult (`Action<DropboxRequestResult<byte[]>>`)	Callback function that receives `DropboxRequestResult<byte[]>` that contains result `data`, error `bool` indicator and `errorDescription` if there was any error.
onProgress (`Action<float> onProgress`)	Callback function that receives progress as float from 0 to 1.
useCachedFirst (`bool`)	Use cached version if it exists?
useCachedIfOffline (`bool`)	Use cached version if no Internet connection?
receiveUpdates (`bool`)	If `true`, then when there are remote updates on Dropbox, callback function `onResult` will be triggered again with updated version of the file.

Example usage

```
// Download file from Dropbox (or used cached if no updates) as byte array.
DropboxSync.Main.GetFileAsBytes("/DropboxSyncExampleFolder/image.jpg", (res) => {
    if(res.error != null){
        Debug.LogError("Failed to get file bytes: "+res.error.ErrorDescription);
    }else{
        var imageBytes = res.data;
        Debug.Log("Got file as bytes array, length: "
                                            +imageBytes.Length.ToString()+" bytes");
    }
}, receiveUpdates:true);
```

GetFile<T>()

```
public void GetFile<T>(string dropboxPath,
                       Action<DropboxRequestResult<T>> onResult,
                       Action<float> onProgress = null,
                       bool useCachedFirst = false,
                       bool useCachedIfOffline = true,
                       bool receiveUpdates = false)
```

Asynchronously retrieves file from Dropbox and tries to produce object of specified type `T`.

Supported generic types `T`:

- `string` - for text files
- `JsonObject`, `JsonArray` - for JSON-formatted files
- `Texture2D` - for image files

Parameters

Parameter	Description
dropboxPath (`string`)	Path to file on Dropbox or inside of Dropbox App folder (depending on accessToken type). Should start with "/". *Example: /DropboxSyncExampleFolder/image.jpg*
onResult (`Action<DropboxRequestResult<T>>`)	Callback function that receives `DropboxRequestResult<T>` that contains result `data`, error `bool` indicator and `errorDescription` if there was any error.
onProgress (`Action<float> onProgress`)	Callback function that receives progress as float from 0 to 1.
useCachedFirst (`bool`)	Use cached version if it exists?
useCachedIfOffline (`bool`)	Use cached version if no Internet connection?
receiveUpdates (`bool`)	If `true`, then when there are remote updates on Dropbox, callback function `onResult` will be triggered again with updated version of the file.

Example usage

```
// Download Jpeg image from Dropbox (or used cached if no updates)
// and receive it as Texture2D object.
DropboxSync.Main.GetFile<Texture2D>("/DropboxSyncExampleFolder/image.jpg", (res) => {
    if(res.error != null){
        Debug.LogError("Error getting picture from Dropbox: "
                                            +res.error.ErrorDescription);
    }else{
        Debug.Log("Received picture from Dropbox!");
        var tex = res.data;
        UpdatePicture(tex);
    }
}, useCachedFirst:true);
```

GetFileAsLocalCachedPath()

```
public void GetFileAsLocalCachedPath(string dropboxPath,
                                     Action<DropboxRequestResult<string>> onResult,
                                     Action<float> onProgress = null,
                                     bool useCachedFirst = false,
                                     bool useCachedIfOffline = true,
                                     bool receiveUpdates = false)
```

Asynchronously retrieves file from Dropbox and returns path to local filesystem cached copy.

Parameters

Parameter	Description
dropboxPath (`string`)	Path to file on Dropbox or inside of Dropbox App folder (depending on accessToken type). Should start with "/". *Example: /DropboxSyncExampleFolder/image.jpg*
onResult (`Action<DropboxRequestResult<string>>`)	Callback function that receives `DropboxRequestResult<string>` that contains result `data`, error `bool` indicator and `errorDescription` if there was any error.
onProgress (`Action<float> onProgress`)	Callback function that receives progress as float from 0 to 1.
useCachedFirst (`bool`)	Use cached version if it exists?
useCachedIfOffline (`bool`)	Use cached version if no Internet connection?
receiveUpdates (`bool`)	If `true`, then when there are remote updates on Dropbox, callback function `onResult` will be triggered again with updated version of the file.

Example usage

```
// Download video from Dropbox (or use cached if no updates)
// and get local filepath.
DropboxSync.Main.GetFileAsLocalCachedPath("/DropboxSyncExampleFolder/video.mp4",
 (res) => {
    if(res.error != null){
        Debug.LogError("Error getting video from Dropbox: "
                                            +res.error.ErrorDescription);
    }else{
        Debug.Log("Received video from Dropbox!");
        var filePathInCache = res.data;
        PlayVideo(filePathInCache);
    }
}, receiveUpdates:true);
```

`Uploading files`

UploadFile(localFilePath)

```
public void UploadFile(string dropboxPath,
                       string localFilePath,
                       Action<DropboxRequestResult<DBXFile>> onResult,
                       Action<float> onProgress = null)
```

Uploads file from specified filepath in local filesystem to Dropbox.

Parameters

Parameter	Description
dropboxPath (`string`)	Dropbox path where to upload file. *Example: /my_text.txt*
localFilePath (`string`)	Full file path in local filesystem. *Example: C:/my_text.txt*
onResult (`Action<DropboxRequestResult<DBXFile>>`)	Result callback that receives created remote file metadata
onProgress (`Action<float> onProgress`)	Callback function that receives progress as float from 0 to 1.

Example usage

```
DropboxSync.Main.UploadFile("/my_text.txt", "C:/my_text.txt", (res) => {
    if(res.error != null){
        Debug.LogError("Error uploading file: "+res.error.ErrorDescription);
    }else{
        Debug.Log("File uploaded!");
    }
}, (progress) => {
    Debug.Log("Uploading file... "+Mathf.RoundToInt(progress*100)+"%");
});
```

UploadFile(bytes[])

```
public void UploadFile(string dropboxPath,
                       byte[] bytes,
                       Action<DropboxRequestResult<DBXFile>> onResult,
                       Action<float> onProgress = null)
```

Uploads byte[] to specified Dropbox path.

Parameters

Parameter	Description
dropboxPath (`string`)	Dropbox path where to upload file. *Example: /my_text.txt*
bytes (`byte[]`)	Bytes array containing file data
onResult (`Action<DropboxRequestResult<DBXFile>>`)	Result callback that receives created remote file metadata
onProgress (`Action<float> onProgress`)	Callback function that receives progress as float from 0 to 1.

Example usage

```
var bytes = Encoding.UTF8.GetBytes("my text");

DropboxSync.Main.UploadFile("/my_text.txt", bytes, (res) => {
    if(res.error != null){
        Debug.LogError("Error uploading file: "+res.error.ErrorDescription);
    }else{
        Debug.Log("File uploaded!");
    }
}, (progress) => {
    Debug.Log("Uploading file... "+Mathf.RoundToInt(progress*100)+"%");
});
```

`Getting folder structure`

GetFolderStructure()

```
public void GetFolderStructure(string dropboxFolderPath,
                               Action<DropboxRequestResult<DBXFolder>> onResult,
                               Action<float> onProgress = null)
```

Asynchronously retrieves DBXFolder object with all recursive sub structure.

Parameters

Parameter	Description
dropboxFolderPath (`string`)	Path to folder on Dropbox or inside Dropbox App (depending on accessToken type). Should start with "/". *Example: /DropboxSyncExampleFolder*
onResult (`Action<DropboxRequestResult<DBXFolder>>`)	Callback function that receives `DropboxRequestResult<DBXFolder>` that contains result `data` of type `DBXFolder`, error `bool` indicator and `errorDescription` if there was any error.
onProgress (`Action<float> onProgress`)	Callback function that receives progress as float from 0 to 1.

GetFolderItems()

```
public void GetFolderItems(string path,
                           Action<DropboxRequestResult<List<DBXItem>>> onResult,
                           Action<float> onProgress = null,
                           bool recursive = false)
```

Asynchronously retrieves flat list of `DBXItem` objects (files and folders).

Parameters

Parameter	Description
dropboxFolderPath (`string`)	Path to folder on Dropbox or inside Dropbox App (depending on accessToken type). Should start with "/". *Example: /DropboxSyncExampleFolder*
onResult (`Action<DropboxRequestResult<List<DBXItem>>>`)	Callback function that receives `DropboxRequestResult<List<DBXItem>>` that contains result `data` of type `List<DBXItem>`, error `bool` indicator and `errorDescription` if there was any error.
onProgress (`Action<float> onProgress`)	Callback function that receives progress as float from 0 to 1.
recursive (`bool`)	If `true`, then resulting list contains all subfolders and their files recursively.

Example usage

```
// Count all files on Dropbox
DropboxSync.Main.GetFolderItems("/", (res) => {
    if(res.error != null){
        Debug.LogError("Failed to get folder items for folder "
                    +dropboxFolderPath+" "+res.error.ErrorDescription);
    }else{
        Debug.Log("Total files on Dropbox: "
            +(res.data.Where(x => x.type == DBXItemType.File).Count()).ToString());
    }
}, recursive: true);
```

`Changes notifications`

SubscribeToFileChanges()

```
public void SubscribeToFileChanges(string dropboxFilePath,
                                   Action<DBXFileChange> onChange)
```

Subscribes to file changes on Dropbox.

Callback fires once, when change is being registered and changed file checksum is cached in local metadata. If change was made not during app runtime, callback fires as soon as app is running and checking for updates.

Update interval can be changed using `DBXChangeForChangesIntervalSeconds` (default values if 5 seconds).

Parameters

Parameter	Description
dropboxFilePath (`string`)	Path to file on Dropbox or inside Dropbox App (depending on accessToken type). Should start with "/". *Example: /DropboxSyncExampleFolder/image.jpg*
onChange (`Action<DBXFileChange>`)	Callback function that receives `DBXFileChange` that contains `changeType` and `DBXFile` (updated file metadata).

SubscribeToFolderChanges()

```
public void SubscribeToFolderChanges(string dropboxFilePath,
                                     Action<List<DBXFileChange>> onChange)
```

Subscribes to file changes on Dropbox in specified folder (and recursively to all subfolders and their files).

Callback fires once, when change is being registered and changed file checksum is cached in local metadata. If change was made not during app runtime, callback fires as soon as app is running and checking for updates.

Update interval can be changed using `DBXChangeForChangesIntervalSeconds` (default values if 5 seconds).

Parameters

Parameter	Description
dropboxFolderPath (`string`)	Path to folder on Dropbox or inside Dropbox App (depending on accessToken type). Should start with "/". *Example: /DropboxSyncExampleFolder*
onChange (`Action<List<DBXFileChange>>`)	Callback function that receives `List<DBXFileChange>` consisting of file changes. Each file change contains `changeType` and `DBXFile` (updated file metadata).

UnsubscribeFromChangesOnPath()

```
public void UnsubscribeFromChangesOnPath(string dropboxPath,
                                          Action<List<DBXFileChange>> onChange)
```

Unsubscribe specific callback from changes on specified dropbox path

Parameters

Parameter	Description
dropboxPath (`string`)	Path from which to unsubscribe
onChange (`Action<List<DBXFileChange>>`)	Callback reference

UnsubscribeAllFromChangesOnPath()

```
public void UnsubscribeAllFromChangesOnPath(string dropboxPath)
```

Unsubscribes all subscribers from changes on specified Dropbox path

Parameters

Parameter	Description
dropboxPath (`string`)	Path from which to unsubscribe

`Other`

PathExists()

```
public void PathExists(string dropboxPath,
                       Action<DropboxRequestResult<bool>> onResult)
```

Checks if dropbox path (file or folder) exists

Parameters

Parameter	Description
dropboxPath (`string`)	Path to file or folder on Dropbox or inside of Dropbox App folder (depending on accessToken type). Should start with "/". *Example: /DropboxSyncExampleFolder/image.jpg*
onResult (`Action<DropboxRequestResult<bool>>`)	Result callback containing bool that indicates existance on the item

Example usage

```
DropboxSync.Main.PathExists("/my_text.txt", (res) => {
    if(res.error != null){
        Debug.LogError("Failed to check if file exists: "
                       +res.error.ErrorDescription);
    }else{
        if(res.data){
            // file exists
            Debug.Log("My text file exists!");
        }else{
            Debug.Log("File wasn't found :(");
        }
    }
});
```

Move()

```
public void Move(string dropboxFromPath,
                 string dropboxToPath,
                 Action<DropboxRequestResult<DBXItem>> onResult)
```

Moves file or folder from dropboxFromPath to dropboxToPath

Parameters

Parameter	Description
dropboxFromPath (`string`)	From path
dropboxToPath (`string`)	To path
onResult (`Action<DropboxRequestResult<DBXItem>>`)	Result callback containing metadata of moved object

Example usage

```
DropboxSync.Main.Move("/my_text.txt", "/my_text_renamed.txt", (res) => {
    if(res.error != null){
        Debug.LogError("Failed to move file: "+res.error.ErrorDescription);
    }else{
        Debug.Log("Successfully moved file!");
    }
});
```

Delete()

```
public void Delete(string dropboxPath,
                   Action<DropboxRequestResult<DBXItem>> onResult)
```

Deletes file or folder on Dropbox

Parameters

Parameter	Description
dropboxPath (`string`)	Path to file or folder on Dropbox or inside of Dropbox App folder (depending on accessToken type). Should start with "/". *Example: /DropboxSyncExampleFolder/image.jpg*
onResult (`Action<DropboxRequestResult<DBXItem>>`)	Callback function that receives DropboxRequestResult with DBXItem metadata of deleted file or folder

Example usage

```
DropboxSync.Main.Delete("/my_text.txt", (res) => {
    if(res.error != null){
        Debug.LogError("Failed to delete file: "+res.error.ErrorDescription);
    }else{
        Debug.Log("Successfully deleted file!");
    }
});
```

CreateFolder()

```
public void CreateFolder(string dropboxFolderPath,
                         Action<DropboxRequestResult<DBXFolder>> onResult)
```

Creates folder using path specified. If nested in non-existing parent folders - creates this parent folders as well.

Parameters

Parameter	Description
dropboxFolderPath (`string`)	Path of folder to create
onResult (`Action<DropboxRequestResult<DBXFolder>>`)	Result callback that contains metadata of the created folder

Example usage

```
DropboxSync.Main.CreateFolder("/folder/test/nested", (res) => {
    if(res.error != null){
        Debug.LogError("Failed to create folder: "+res.error.ErrorDescription);
    }else{
        Debug.Log("Successfully created folder!");
    }
});
```