
dbox (THIS LIBRARY IS V1 ONLY WHICH IS NOW OBSOLETE!!)

A Node.JS convenience wrapper around the Dropbox API. Simplifies OAuth handshake and removes HTTP ceremony.

Installation

I always recommend you bundle your dependencies with your application. To do this, create a **package.json** file in the root of your project with the minimum information...

```
1 {
2   "name": "yourapplication",
3   "version": "0.1.0",
4   "dependencies": {
5     "dbox": "0.6.1"
6   }
7 }
```

Then run the following command using npm...

```
1 npm install
```

OR, if you just want to start playing with the library run...

```
1 npm install dbox
```

API Overview

dbox methods (where **dbox** is set from requiring the **dbox** library)...

```
1 app          <-- creates application object
```

app methods (where **app** is created from the above **app** call)...

```
1 requesttoken  <-- creates request token for getting request token
    and authorization url
2 accesstoken  <-- creates access token for creating a client
    object
3 client        <-- creates client object with access to users
    dropbox account
```

client methods (where **client** is created from the above **client** call)...

```
1 account      <-- view account
2 mkdir        <-- make directory
```

```
3 mv          <-- move file or directory
4 cp          <-- copy file or directory
5 rm          <-- remove file or directory
6 put         <-- upload file
7 get         <-- download file
8 metadata    <-- get file or directory information
9 revisions   <-- get revision history
10 restore     <-- restore previous version
11 search     <-- search directory
12 shares     <-- create link to view file
13 media      <-- create streamable link to file
14 thumbnails <-- get thumbnail of file
15 copyref    <-- create copy reference to file
16 delta      <-- get list of delta entries
17 stream     <-- creates readable stream
18 readdir    <-- recursively reads directory
```

How to Use

Creating a functional `dbox` client is a four step process.

1. create an `app` using application credentials provided by dropbox
2. obtain request token to use for generation access token
3. have user visit authorization URL to grant access to your application
4. create a client using access token that was generated earlier

Step 1

```
1 var dbox = require("dbox")
2 var app  = dbox.app({ "app_key": "umdez34678ck01fx", "app_secret": "
    tjm89017sci88o6" })
```

Step 2

Authorization is a three step process.

- a) Get a request token...

```
app.requesttoken(function(status, request_token){ console.log(request_token) })
```

- b) User must visit the url to grant authorization to the client...

```
https://www.dropbox.com/1/oauth/authorize?oauth_token=#{ request_token.oauth_token }
```

c) Generate our access token with the request token...

```
app.accesstoken(request_token, function(status, access_token){ console.log(access_token) })
```

Step 3

```
1 var client = app.client(access_token)
```

Now we have a client that gives us access to all the api functionality.

Client Methods

account([options,] callback)

Returns account information.

```
1 client.account(function(status, reply){
2   console.log(reply)
3 })
```

output of `reply` returns...

```
1 {
2   uid: 123456789,
3   display_name: 'Brock Whitten',
4   email: 'brock@sintaxi.com',
5   country: 'CA',
6   referral_link: 'https://www.dropbox.com/referrals/NTc0NzYwNDc5',
7   quota_info: {
8     shared: 1100727791,
9     quota: 2415919104,
10    normal: 226168599
11  }
12 }
```

mkdir(path, [options,] callback)

Creates directory at specified location.

```
1 client.mkdir("foo", options, function(status, reply){
2   console.log(reply)
3 })
```

output of `reply` returns...

```
1 {
2   "size": "0 bytes",
3   "rev": "1f477dd351f",
4   "thumb_exists": false,
5   "bytes": 0,
6   "modified": "Wed, 10 Aug 2011 18:21:30 +0000",
7   "path": "/foo",
8   "is_dir": true,
9   "icon": "folder",
10  "root": "sandbox",
11  "revision": 5023410
12 }
```

mv(from_path, to_path, [options,] callback)

Moves file or directory to a new location.

```
1 client.mv("foo", "bar", function(status, reply){
2   console.log(reply)
3 })
```

output of `reply` returns...

```
1 {
2   "size": "0 bytes",
3   "rev": "irt77dd3728",
4   "thumb_exists": false,
5   "bytes": 0,
6   "modified": "Wed, 10 Aug 2011 18:21:30 +0000",
7   "path": "/bar",
8   "is_dir": true,
9   "icon": "folder",
10  "root": "sandbox",
11  "revision": 5023410
12 }
```

cp(from_path, to_path, [options,] callback)

Copies a file or directory to a new location.

```
1 client.cp("bar", "baz", function(status, reply){
2   console.log(reply)
3 })
4
5 {
6   "size": "0 bytes",
```

```
7   "rev": "irt77dd3728",
8   "thumb_exists": false,
9   "bytes": 0,
10  "modified": "Wed, 10 Aug 2011 18:21:30 +0000",
11  "path": "/baz",
12  "is_dir": true,
13  "icon": "folder",
14  "root": "sandbox",
15  "revision": 5023410
16 }
```

rm(path, [options,] callback)

Removes a file or directory.

```
1 client.rm("README.txt", function(status, reply){
2   console.log(reply)
3 })
```

output of `reply` returns...

```
1 {
2   "size": "0 bytes",
3   "is_deleted": true,
4   "bytes": 0,
5   "thumb_exists": false,
6   "rev": "1f33043551f",
7   "modified": "Wed, 10 Aug 2011 18:21:30 +0000",
8   "path": "/README.txt",
9   "is_dir": false,
10  "icon": "page_white_text",
11  "root": "sandbox",
12  "mime_type": "text/plain",
13  "revision": 492341
14 }
```

put(path, data, [options,] callback)

Creates or modifies a file with given data. `data` may be a string or a buffer.

```
1 client.put("foo/hello.txt", "here is some text", function(status, reply)
2   ){
3   console.log(reply)
4 }
```

output of `reply` returns...

```
1 {
2   "size": "225.4KB",
3   "rev": "35e97029684fe",
4   "thumb_exists": false,
5   "bytes": 230783,
6   "modified": "Tue, 19 Jul 2011 21:55:38 +0000",
7   "path": "/foo/hello.txt",
8   "is_dir": false,
9   "icon": "page_white_text",
10  "root": "sandbox",
11  "mime_type": "text/plain",
12  "revision": 220823
13 }
```

get(path, [options,] callback)

Pulls down file (available as a buffer) with its metadata.

```
1 client.get("foo/hello.txt", function(status, reply, metadata){
2   console.log(reply.toString(), metadata)
3 })
```

output of `reply.toString()` returns...

```
1 here is some text
```

output of `metadata` returns...

```
1 {
2   "revision": 11,
3   "rev": "b07a93bb3",
4   "thumb_exists": false,
5   "bytes": 17,
6   "modified": "Sat, 12 May 2012 19:31:08 +0000",
7   "client_mtime": "Sat, 12 May 2012 19:30:52 +0000",
8   "path": "/foo/hello.txt",
9   "is_dir": false,
10  "icon": "page_white_text",
11  "root": "app_folder",
12  "mime_type": "text/plain",
13  "size": "17 bytes"
14 }
```

metadata(path, [options,] callback)

Retrieves file or directory metadata.

```
1 // available options...
2 var options = {
3   file_limit      : 10000,           // optional
4   hash            : ...,            // optional
5   list            : true,           // optional
6   include_deleted : false,          // optional
7   rev             : 7,              // optional
8   locale:         : "en",           // optional
9   root:           : "sandbox"       // optional
10 }
11
12 client.metadata("Getting_Started.pdf", options, function(status, reply)
13   {
14     console.log(reply)
15   })
```

output of `reply` returns...

```
1 {
2   "size": "225.4KB",
3   "rev": "35e97029684fe",
4   "thumb_exists": false,
5   "bytes": 230783,
6   "modified": "Tue, 19 Jul 2011 21:55:38 +0000",
7   "path": "/Getting_Started.pdf",
8   "is_dir": false,
9   "icon": "page_white_acrobat",
10  "root": "sandbox",
11  "mime_type": "application/pdf",
12  "revision": 220823
13 }
```

revisions(path, [options,] callback)

Obtains metadata for the previous revisions of a file.

```
1 // available options...
2 var options = {
3   rev_limit      : 10,              // optional
4   locale:        : "en"            // optional
5 }
6
7 client.revisions("foo/hello.txt", options, function(status, reply){
8   console.log(reply)
9 })
```

output of `reply` returns...

```
1  [
2    {
3      "is_deleted": true,
4      "revision": 4,
5      "rev": "400000000d",
6      "thumb_exists": false,
7      "bytes": 0,
8      "modified": "Wed, 20 Jul 2011 22:41:09 +0000",
9      "path": "foo/hello.txt",
10     "is_dir": false,
11     "icon": "page_white",
12     "root": "sandbox",
13     "mime_type": "text/plain",
14     "size": "0 bytes"
15   },
16   {
17     "revision": 1,
18     "rev": "100000000d",
19     "thumb_exists": false,
20     "bytes": 3,
21     "modified": "Wed, 20 Jul 2011 22:40:43 +0000",
22     "path": "foo/hello.txt",
23     "is_dir": false,
24     "icon": "page_white",
25     "root": "sandbox",
26     "mime_type": "text/plain",
27     "size": "3 bytes"
28   }
29 ]
```

restore(path, rev, [options,] callback)

Restores a file path to a previous revision.

```
1  client.revisions("foo/hello.txt", 4, function(status, reply){
2    console.log(reply)
3  })
```

output of `reply` returns...

```
1  {
2    "is_deleted": true,
3    "revision": 4,
4    "rev": "400000000d",
5    "thumb_exists": false,
6    "bytes": 0,
7    "modified": "Wed, 20 Jul 2011 22:41:09 +0000",
8    "path": "/foo/hello.txt",
9    "is_dir": false,
```

```
10  "icon": "page_white",
11  "root": "sandbox",
12  "mime_type": "text/plain",
13  "size": "0 bytes"
14 }
```

search(path, query, [options,] callback)

Returns metadata for all files and directories that match the search query.

```
1  var options = {
2    file_limit      : 10000,           // optional
3    include_deleted : false,           // optional
4    locale:         : "en"             // optional
5  }
6
7  client.search("foo", "hello", options, function(status, reply){
8    console.log(reply)
9  })
```

output of `reply` returns...

```
1  [
2    {
3      "size": "0 bytes",
4      "rev": "35c1f029684fe",
5      "thumb_exists": false,
6      "bytes": 0,
7      "modified": "Mon, 18 Jul 2011 20:13:43 +0000",
8      "path": "/foo/hello.txt",
9      "is_dir": false,
10     "icon": "page_white_text",
11     "root": "sandbox",
12     "mime_type": "text/plain",
13     "revision": 220191
14   }
15 ]
```

shares(path, [options,] callback)

Creates and/or returns a shareable link to a file or directory.

```
1  client.shares("foo/hello.txt", options, function(status, reply){
2    console.log(reply)
3  })
```

output of `reply` returns...

```
1 {
2   "url": "http://db.tt/APqhX1",
3   "expires": "Sat, 17 Aug 2011 02:34:33 +0000"
4 }
```

media(path, [options,] callback)

Creates and/or returns a shareable link to a file or directory. This endpoint is similar to /shares but content is streamable.

```
1 client.media("foo/hello.txt", function(status, reply){
2   console.log(reply)
3 })
```

output of `reply` returns...

```
1 {
2   "url": "http://www.dropbox.com/s/m/a2mbDa2",
3   "expires": "Thu, 16 Sep 2011 01:01:25 +0000"
4 }
```

thumbnails(path, [options,] callback)

Gets a thumbnail for an image.

```
1 client.thumbnails("foo/koala.jpg", function(status, reply, metadata){
2   console.log(metadata);
3
4   require('fs').writeFile('koala_small.jpg', reply, function () {
5     console.log('Thumbnail saved!');
6   });
7 })
```

output of `reply` is a buffer which should be sent to a new image file.

output of `metadata` returns...

```
1 {
2   "revision": 13,
3   "rev": "d07a93bb3",
4   "thumb_exists": true,
5   "bytes": 780831,
6   "modified": "Sat, 12 May 2012 19:48:59 +0000",
7   "client_mtime": "Tue, 14 Jul 2009 05:32:31 +0000",
8   "path": "/foo/koala.jpg",
9   "is_dir": false,
```

```
10  "icon": "page_white_picture",
11  "root": "app_folder",
12  "mime_type": "image/jpeg",
13  "size": "762.5 KB"
14 }
```

cpref(path, [options,] callback)

```
1  client.cpref("song.mp3", function(status, reply){
2    console.log(reply)
3  })
```

output of `reply` returns...

```
1  {
2    expires: 'Thu, 03 Apr 2042 22:33:49 +0000',
3    copy_ref: 'ALGf72Jrc3A0ZTh5MzA4Mg'
4  }
```

delta([options,] callback)

```
1  client.delta(function(status, reply){
2    console.log(reply)
3  })
```

output of `reply` returns...

```
1  {
2    reset: true,
3    cursor: '
      AkMCE0f1CsMA7tobhXR1vwEZaM1KjFqTNjxgWITCks6oeJxjKBL2Z2Co0W0p_rS0gYHxJwMQAAyKw
      ',
4    has_more: false,
5    entries: [
6      [ '/foo', [Object] ],
7      [ '/bar', [Object] ]
8    ]
9  }
```

readdir(path, callback)

Get an array of paths for all files and directories found in the given path. The method calls recursively to dropbox so it can take a long time to evaluate.

```
1 client.readdir('/', function(status, reply){
2     console.log(reply)
3 })
```

Output of `readdir` returns...

```
1 ['/', '/foo', '/bar']
```

License

Copyright 2011 Chloi Inc. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.