Core API

The Core API is the underlying interface for all of our official Dropbox mobile apps and our SDKs. It's the most direct way to access the API. This reference document is designed for those interested in developing for platforms not supported by the SDKs or for those interested in exploring API features in detail.

General notes

API compatibility

This API will evolve. Future versions of this API may add new endpoints or parameters. In order to keep older clients working, the behavior and return value of APIs with given parameter values will not change from the currently documented behavior and return values, with two important exceptions: currently undocumented request parameters (whether they are actually ignored or not) may be given a specific meaning, and objects returned in responses may contain additional keys in the future.

Thus, clients that want to be future-proof should avoid passing undocumented parameters (as they may cause different behavior in the future), and they should avoid strict checks on the keys of objects found in responses.

For example, you should not consider a /metadata response invalid if it contains additional keys besides those currently documented below.

SSL only

We require that all requests are done over SSL.

App folder access type

The default root level access type, **app folder** (as described in app types and permissions), is referenced in API URLs by its codename `sandbox`. This is the only place where such a distinction is made.

UTF-8 encoding

Every string passed to and from the Dropbox API needs to be UTF-8 encoded. For maximum compatibility, normalize to Unicode Normalization Form C (NFC) before UTF-8 encoding.

Date format

All dates in the API are strings in the following format:

```
"Sat, 21 Aug 2010 22:31:20 +0000"
```

In code format, which can be used in all programming languages that support `strftime` or `strptime`:

```
"%a, %d %b %Y %H:%M:%S %z"
```

Locale global request parameter

Dropbox uses the `locale` parameter to specify language settings of content responses. If your app supports any language other than English, insert the appropriate IETF language tag. When a supported language is specified, Dropbox will returned translated `size` and/or `user_error` fields (where applicable) .

Error handling

Errors are returned using standard HTTP error code syntax. Any additional info is included in the body of the return call, JSON-formatted. Error codes not listed here are in the REST API methods listed below.

Standard API errors

Code	Description
400	Bad input parameter. Error message should indicate which one and why.
401	Bad or expired token. This can happen if the user or Dropbox revoked or expired an access token. To fix, you should re-authenticate the user.
403	Bad OAuth request (wrong consumer key, bad nonce, expired timestamp...). Unfortunately, re-authenticating the user won't help here.
404	File or folder not found at the specified path.
405	Request method not expected (generally should be GET or POST).
429	Your app is making too many requests and is being rate limited. 429s can trigger on a per-app or per-user basis.
503	If the response includes the `Retry-After` header, this means your OAuth 1.0 app is being rate limited. Otherwise, this indicates a transient server error, and your app should retry its request.
507	User is over Dropbox storage quota.
5xx	Server error. Check DropboxOps.

OAuth 1.0

OAuth 1.0 continues to be supported for all API requests, but OAuth 2.0 is now preferred.

/oauth/request_token

Description

Step 1 of authentication. Obtain an OAuth *request token* to be used for the rest of the authentication process.

This method corresponds to Obtaining an Unauthorized Request Token in the OAuth Core 1.0 specification.

URL Structure

```
https://api.dropbox.com/1/oauth/request_token
```

Method

POST

Parameters

There are no Dropbox-specific parameters for this method. See Consumer Obtains a Request Token in the OAuth Core 1.0 specification for a description of the OAuth parameters used for fetching a request token. Since this method is on behalf of an unauthenticated user, no access token or secret should be involved when signing or sending the request.

Returns

A request token and the corresponding request token secret, URL-encoded. This token/secret pair is meant to be used with `/oauth/access_token` to complete the authentication process and cannot be

used for any other API calls. See Service Provider Issues an Unauthorized Request Token in the OAuth Core 1.0 specification for additional discussion of the values returned when fetching a request token.

Sample response

```
oauth_token_secret=b9q1n5il4lcc&oauth_token=mh7an9dkrg59
```

/oauth/authorize

Description

Step 2 of authentication. Applications should direct the user to `/oauth/authorize`. This isn't an API call per se, but rather a web endpoint that lets the user sign in to Dropbox and choose whether to grant the application the ability to access files on their behalf. Without the user's authorization in this step, it isn't possible for your application to obtain an access token from `/oauth/access_token`.

This method corresponds to Obtaining User Authorization in the OAuth Core 1.0 specification.

URL Structure

```
https://www.dropbox.com/1/oauth/authorize
```

Note: This is the only step that requires an endpoint on `www.dropbox.com`. All other API requests are done via `api.dropbox.com`, `api-content.dropbox.com`, or `api-notify.dropbox.com`. The user must be redirected to the Dropbox website over HTTPS. After the user decides whether or not to authorize your application, they will be redirected to the URL specified by `oauth_callback` (more on that below).

Method

GET

Parameters

oauth_token *required* The request token obtained via `/oauth/request_token`.

oauth_callback After the user either authorizes or disallows your application, they are redirected to this URL.

locale If the locale specified is a supported language, Dropbox will direct users to a translated version of the authorization website. See the notes above for more information about supported locales.

disable_signup When true (default is false) users will not be able to sign up for a Dropbox account via the authorization page. Instead, the authorization page will show a link to the Dropbox iOS app in the App Store. This is only intended for use when necessary for compliance with App Store policies.

Returns

Because the application doesn't call `/oauth/authorize` directly, there is no direct return value. After the user authorizes the application, the request token can be used to retrieve an access token via the `/oauth/access_token` API call. If the `oauth_callback` parameter is omitted, the application must find some other way of determining when the authorization step is complete. For example, the application can have the user explicitly indicate to it that this step is complete, but this flow may be less intuitive for users than the redirect flow.

If `oauth_callback` is specified and the user authorizes the application, they will get redirected to the specified URL with the following additional URL query parameters appended:

oauth_token The request token that was just authorized. The request token secret isn't sent back.

uid The user's unique Dropbox ID.

If the user chooses not to authorize the application, they will get redirected to the `oauth_callback` URL with the additional URL query parameter **not_approved=true**.

/oauth/access_token

Description
Step 3 of authentication. After the `/oauth/authorize` step is complete, the application can call `/oauth/access_token` to acquire an access token.

This method corresponds to Obtaining an Access Token in the OAuth Core 1.0 specification.

URL Structure

```
https://api.dropbox.com/1/oauth/access_token
```

Method
POST
Parameters
There are no Dropbox-specific parameters for this method. See Consumer Requests an Access Token in the OAuth Core 1.0 specification for a description of the parameters used for fetching an access token. Note that the `oauth_token` and `oauth_token_secret` for this method are the request token and request token secret obtained previously via `/oauth/request_token`.

Returns
URL-encoded access token, access token secret, and Dropbox user id. Upon return, the authorization process is now complete and the access token and corresponding secret can be used to sign requests for the main API calls. See Service Provider Grants an Access Token in the OAuth Core 1.0 specification for additional discussion of the values returned when fetching an access token. If your app is configured to work within an app folder, that folder is also created during this step.

Sample response

```
oauth_token_secret=95grkd9na7hm&oauth_token=ccl4li5n1q9b&uid=100
```

OAuth 2.0

Dropbox supports OAuth 2.0 for authenticating all API requests.

/oauth2/authorize
Python
Java
Ruby
PHP

Description

This starts the OAuth 2.0 authorization flow. This isn't an API call—it's the web page that lets the user sign in to Dropbox and authorize your app. After the user decides whether or not to authorize your app, they will be redirected to the URI specified by `redirect_uri`.

OAuth 2.0 supports two authorization flows:

The `code` flow returns a code via the `redirect_uri` callback which should then be converted into a bearer token using the `/oauth2/token` call. This is the recommended flow for apps that are running on a server.

The `token` or implicit grant flow returns the bearer token via the `redirect_uri` callback, rather than requiring your app to make a second call to a server. This is useful for pure client-side apps, such as mobile apps or JavaScript-based apps.

For more information on the two flows, see Section 1.3 of the OAuth 2 spec.

URL Structure

```
https://www.dropbox.com/1/oauth2/authorize
```

Note: This is the only step that requires an endpoint on `www.dropbox.com`. All other API requests are done via `api.dropbox.com`, `api-content.dropbox.com`, or `api-notify.dropbox.com`.

Method
GET
Parameters

response_type *required* The grant type requested, either `token` or `code`.

client_id *required* The app's key, found in the App Console.

redirect_uri Where to redirect the user after authorization has completed. This must be the exact URI registered in the App Console; even 'localhost' must be listed if it is used for testing. A redirect URI is required for a `token` flow, but optional for `code`. If the redirect URI is omitted, the `code` will be presented directly to the user and they will be invited to enter the information in your app.

state Up to 200 bytes of arbitrary data that will be passed back to your redirect URI. This parameter should be used to protect against cross-site request forgery (CSRF). See Sections 4.4.1.8 and 4.4.2.5 of the OAuth 2.0 threat model spec.

force_reapprove Whether or not to force the user to approve the app again if they've already done so. If `false` (default), a user who has already approved the application may be automatically redirected to the URI specified by `redirect_uri`. If `true`, the user will not be automatically redirected and will have to approve the app again.

disable_signup When true (default is false) users will not be able to sign up for a Dropbox account via the authorization page. Instead, the authorization page will show a link to the Dropbox iOS app in the App Store. This is only intended for use when necessary for compliance with App Store policies.

Returns
Because `/oauth2/authorize` is a website, there is no direct return value. However, after the user authorizes your app, they will be sent to your redirect URI. The type of response varies based on the `response_type`.

Code flow

These parameters are passed in the query string (after the ? in the URL):

code The authorization code, which can be used to attain a bearer token by calling `/oauth2/token`.

state The state content, if any, originally passed to `/oauth2/authorize`.

Sample response

```
[REDIRECT_URI]?code=ABCDEFG&state=[STATE]
```

Token flow

These parameters are passed in the URL fragment (after the # in the URL):

access_token A token which can be used to make calls to the Dropbox API.

token_type The type of token, which will always be `bearer`.

uid The Dropbox user ID of the authorized user.

state The state content, if any, originally passed to `/oauth2/authorize`.

Sample response

```
[REDIRECT_URI]#access_token=ABCDEFG&token_
type=bearer&uid=12345&state=[STATE]
```

Errors

In either flow, if an error occurs, including if the user has chosen not to authorize the app, the following parameters will be included in the redirect URI:

error An error code per Section 4.1.2.1 of the OAuth 2.0 spec.

error_description A user-friendly description of the error that occurred.

state The state content, if any, originally passed to `/oauth2/authorize`.

/oauth2/token

Python
Java
Ruby
PHP

Description

This endpoint only applies to apps using the authorization code flow. An app calls this endpoint to acquire a bearer token once the user has authorized the app.

Calls to `/oauth2/token` need to be authenticated using the apps's key and secret. These can either be passed as POST parameters (see parameters below) or via HTTP basic authentication. If basic authentication is used, the app key should be provided as the username, and the app secret should be provided as the password.

URL Structure

```
https://api.dropbox.com/1/oauth2/token
```

Method

POST

Parameters

code *required* The code acquired by directing users to `/oauth2/authorize?response_type=code`.

grant_type *required* The grant type, which must be `authorization_code`.

client_id If credentials are passed in `POST` parameters, this parameter should be present and should be the app's key (found in the App Console).

client_secret If credentials are passed in `POST` parameters, this parameter should be present and should be the app's secret.

redirect_uri Only used to validate that it matches the original `/oauth2/authorize`, not used to redirect again.

Returns

A JSON-encoded dictionary including an access token (`access_token`), token type (`token_type`), and Dropbox user ID (uid). The token type will always be "bearer".

Sample response

```
{"access_token": "ABCDEFG", "token_type": "bearer", "uid": "12345"}
```

/oauth2/token_from_oauth1

Python
Java
Ruby
PHP

Description

This endpoint should be used by apps transitioning from OAuth 1 to OAuth 2. It will return an OAuth 2 token for the authenticated user.

Calls to `/oauth2/token_from_oauth1` must be authenticated via OAuth 1.

URL Structure

```
https://api.dropbox.com/1/oauth2/token_from_oauth1
```

Method

POST

Returns

A JSON-encoded dictionary including an access token (`access_token`) and a token type (`token_type`). The token type will always be "bearer".

Sample response

```
{"access_token": "ABCDEFG", token_type: "bearer"}
```

Access tokens

/disable_access_token

Python

Java
Ruby
PHP

Description
Disables the access token used to authenticate the call. This method works for OAuth 1 and OAuth 2 tokens.

URL Structure

```
https://api.dropbox.com/1/disable_access_token
```

Method
POST
Returns
An empty JSON dictionary, which indicates success.

Dropbox accounts

/account/info
Python
Java
Ruby
PHP

Description
Retrieves information about the user's account.

URL Structure

```
https://api.dropbox.com/1/account/info
```

Method
GET
Parameters
locale Use to specify language settings for user error messages and other language specific text. See the notes above for more information about supported locales.

Returns
User account information.

Sample JSON response

```
{
    "referral_link": "https://www.dropbox.com/referrals/r1a2n3d4m5s6t7",
    "display_name": "John P. User",
    "uid": 12345678,
    "team": {
        "name": "Acme Inc."
    },
    "country": "US",
    "quota_info": {
        "shared": 253738410565,
```

```
        "quota": 107374182400000,
        "normal": 680031877871
    }
}
```

Return value definitions

field	description
`referral_link`	The user's referral link.
`display_name`	The user's display name.
`uid`	The user's unique Dropbox ID.
`country`	The user's two-letter country code, if available.
`team`	If the user belongs to a team, contains team information. Otherwise, null.
`team/name`	The name of the team the user belongs to.
`quota_info/normal`	The user's used quota outside of shared folders (bytes).
`quota_info/shared`	The user's used quota in shared folders (bytes).
`quota_info/quota`	The user's total quota allocation (bytes).

Files and metadata

/files (GET)

Python
Java
Ruby
PHP

Description

Downloads a file. Note that this call goes to **api-content.dropbox.com** instead of api.dropbox.com.

URL Structure

```
https://api-content.dropbox.com/1/files/<root>/<path>
```

root The root relative to which `path` is specified. Valid values are `sandbox` and `dropbox`.

path The path to the file you want to retrieve.

Method

GET

Parameter

rev The revision of the file to retrieve. This defaults to the most recent revision.

Returns

The specified file's contents at the requested revision.

The HTTP response contains the content metadata in JSON format within an `x-dropbox-metadata` header.

Errors

404 The file wasn't found at the specified path, or wasn't found at the specified `rev`.

Notes

This method also supports HTTP Range Retrieval Requests to allow retrieving partial file contents.

/files_put

Python

Java
Ruby
PHP

Description
Uploads a file using PUT semantics. Note that this call goes to **api-content.dropbox.com** instead of api.dropbox.com.

This method is in most cases simpler to use than /files (POST).

The preferred HTTP method for this call is **PUT**. For compatibility with browser environments, the **POST** HTTP method is also recognized.

Note: Providing a `Content-Length` header set to the size of the uploaded file is required so that the server can verify that it has received the entire file contents.

URL Structure

```
https://api-content.dropbox.com/1/files_put/<root>/<path>?param=val
```

root The root relative to which `path` is specified. Valid values are `sandbox` and `dropbox`.

path The full path to the file you want to write to. This parameter should *not* point to a folder.

param=val The URL-encoded parameters for this request. They cannot be sent in the request body.

Method
PUT, POST
Request body
required The file contents to be uploaded. Since the entire PUT body will be treated as the file, any parameters must be passed as part of the request URL. The request URL should be signed just as you would sign any other OAuth request URL.

Parameters

locale The metadata returned on successful upload will have its `size` field translated based on the given locale.

overwrite This value, either `true` (default) or `false`, determines what happens when there's already a file at the specified path. If `true`, the existing file will be overwritten by the new one. If `false`, the new file will be automatically renamed (for example, `test.txt` might be automatically renamed to `test (1).txt`). The new name can be obtained from the returned metadata.

parent_rev The revision of the file you're editing. If `parent_rev` matches the latest version of the file on the user's Dropbox, that file will be replaced. Otherwise, the new file will be automatically renamed (for example, `test.txt` might be automatically renamed to `test (conflicted copy).txt`). If you specify a revision that doesn't exist, the file won't save (error 400). Get the most recent `rev` by performing a call to /metadata.

Returns
The metadata for the uploaded file. More information on the returned metadata fields are available here.

Sample JSON response

```
{
    "size": "225.4KB",
    "rev": "35e97029684fe",
    "thumb_exists": false,
    "bytes": 230783,
    "modified": "Tue, 19 Jul 2011 21:55:38 +0000",
    "path": "/Getting_Started.pdf",
    "is_dir": false,
    "icon": "page_white_acrobat",
    "root": "dropbox",
    "mime_type": "application/pdf",
    "revision": 220823
}
```


Errors

411 Missing `Content-Length` header (this endpoint doesn't support HTTP chunked transfer encoding).

Notes

`/files_put` has a maximum file size limit of 150 MB and does not support uploads with chunked encoding. To upload larger files, use `/chunked_upload` instead.

/files (POST)

Description

Uploads a file. Note that this call goes to **api-content.dropbox.com** instead of api.dropbox.com.

We recommend you use /files_put instead due to its simpler interface.

URL Structure

```
https://api-content.dropbox.com/1/files/<root>/<path>
```

root The root relative to which `path` is specified. Valid values are `sandbox` and `dropbox`.

path The path to the folder the file should be uploaded to. This parameter should *not* point to a file.

Method

POST

Request body

required The file contents to be uploaded. Since the entire POST body will be treated as the file, any parameters must be passed as part of the request URL. The request URL should be signed just as you would sign any other OAuth request URL.

Parameters

locale The metadata returned on successful upload will have its `size` field translated based on the given locale.

overwrite This value, either `true` (default) or `false`, determines what happens when there's already a file at the specified path. If `true`, the existing file will be overwritten by the new one. If `false`, the new file will be automatically renamed (for example, `test.txt` might be automatically renamed to `test (1).txt`). The new name can be obtained from the returned metadata.

parent_rev The revision of the file you're editing. If `parent_rev` matches the latest version of the file on the user's Dropbox, that file will be replaced. Otherwise, the new file will be automatically

renamed (for example, `test.txt` might be automatically renamed to `test (conflicted copy).txt`). If you specify a revision that doesn't exist, the file won't save (error 400). Get the most recent `rev` by performing a call to /metadata.

Returns

The metadata for the uploaded file. More information on the returned metadata fields are available here.

Sample JSON response

```
{
    "size": "225.4KB",
    "rev": "35e97029684fe",
    "thumb_exists": false,
    "bytes": 230783,
    "modified": "Tue, 19 Jul 2011 21:55:38 +0000",
    "path": "/Getting_Started.pdf",
    "is_dir": false,
    "icon": "page_white_acrobat",
    "root": "dropbox",
    "mime_type": "application/pdf",
    "revision": 220823
}
```

Errors

400 The file extension is on Dropbox's ignore list (e.g. **thumbs.db** or **.ds_store**).
404 The `parent_rev` of the file wasn't found.
411 Chunked encoding was attempted for this upload, but is not supported for this method. (For chunked encoding, use `/chunked_upload` instead.)

Notes

`/files` has a maximum file size limit of 150 MB. To upload larger files, use `/chunked_upload` instead.

/metadata

Python
Java
Ruby
PHP

Description

Retrieves file and folder metadata.

URL Structure

```
https://api.dropbox.com/1/metadata/<root>/<path>
```

root The root relative to which `path` is specified. Valid values are `sandbox` and `dropbox`.

path The path to the file or folder.

Method

GET

Parameters

file_limit Default is 10,000 (max is 25,000). When listing a folder, the service won't report listings containing more than the specified amount of files and will instead respond with a 406 (Not Acceptable) status response.

hash Each call to `/metadata` on a folder will return a `hash` field, generated by hashing all of the metadata contained in that response. On later calls to `/metadata`, you should provide that value via this parameter so that if nothing has changed, the response will be a 304 (Not Modified) status code instead of the full, potentially very large, folder listing. This parameter is ignored if the specified path is associated with a file or if `list=false`. A folder shared between two users will have the same hash for each user.

list The strings `true` and `false` are valid values. `true` is the default. If `true`, the folder's metadata will include a `contents` field with a list of metadata entries for the contents of the folder. If `false`, the `contents` field will be omitted.

include_deleted Only applicable when `list` is set. If this parameter is set to true, then `contents` will include the metadata of deleted children. Note that the target of the metadata call is always returned even when it has been deleted (with `is_deleted` set to true) regardless of this flag.

rev If you include a particular revision number, then only the metadata for that revision will be returned.

locale The metadata returned will have its `size` field translated based on the given locale. For more information see above.

include_media_info If true, each file will include a `photo_info` dictionary for photos and a `video_info` dictionary for videos with additional media info. If the data isn't available yet, the string `pending` will be returned instead of a dictionary.

Returns

The metadata for the file or folder at the given `<path>`. If `<path>` represents a folder and the `list` parameter is `true`, the metadata will also include a listing of metadata for the folder's contents.

Sample JSON return value for a file

```
{
    "size": "225.4KB",
    "rev": "35e97029684fe",
    "thumb_exists": false,
    "bytes": 230783,
    "modified": "Tue, 19 Jul 2011 21:55:38 +0000",
    "client_mtime": "Mon, 18 Jul 2011 18:04:35 +0000",
    "path": "/Getting_Started.pdf",
    "is_dir": false,
    "icon": "page_white_acrobat",
    "root": "dropbox",
    "mime_type": "application/pdf",
    "revision": 220823
}
```

Sample JSON return value for a folder when `list` parameter is set to `true`. If `list` is `false` the `contents` key will simply be omitted from the result.

```
{
    "size": "0 bytes",
    "hash": "37eb1ba1849d4b0fb0b28caf7ef3af52",
```

```
    "bytes": 0,
    "thumb_exists": false,
    "rev": "714f029684fe",
    "modified": "Wed, 27 Apr 2011 22:18:51 +0000",
    "path": "/Photos",
    "is_dir": true,
    "icon": "folder",
    "root": "dropbox",
    "contents": [
        {
            "size": "2.3 MB",
            "rev": "38af1b183490",
            "thumb_exists": true,
            "bytes": 2453963,
            "modified": "Mon, 07 Apr 2014 23:13:16 +0000",
            "client_mtime": "Thu, 29 Aug 2013 01:12:02 +0000",
            "path": "/Photos/flower.jpg",
            "photo_info": {
              "lat_long": [
                37.77256666666666,
                -122.45934166666667
              ],
              "time_taken": "Wed, 28 Aug 2013 18:12:02 +0000"
            },
            "is_dir": false,
            "icon": "page_white_picture",
            "root": "dropbox",
            "mime_type": "image/jpeg",
            "revision": 14511
        }
    ],
    "revision": 29007
}
```


Return value definitions

field	description
`size`	A human-readable description of the file size (translated by `locale`).
`bytes`	The file size in bytes.
`path`	Returns the canonical path to the file or directory.
`is_dir`	Whether the given entry is a folder or not.
`is_deleted`	Whether the given entry is deleted (only included if deleted files are being returned).
`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.
`hash`	A folder's `hash` is useful for indicating changes to the folder's contents in later calls to `/metadata`. This is roughly the folder equivalent to a file's `rev`.
`thumb_exists`	True if the file is an image that can be converted to a thumbnail via the `/thumbnails` call.
`photo_info`	Only returned when the `include_media_info` parameter is true and the file is an image. A dictionary that includes the creation time (`time_taken`) and the GPS coordinates (`lat_long`).
`video_info`	Only returned when the `include_media_info` parameter is true and the file is a video. A dictionary that includes the creation time (`time_taken`), the GPS coordinates (`lat_long`), and the length of the video in milliseconds (`duration`).
`icon`	The name of the icon used to illustrate the file type in Dropbox's icon library.
`modified`	The last time the file was modified on Dropbox, in the standard date format (not included for the root folder). For files, this is the modification time set by the desktop client when the file was added to Dropbox, in the standard date format. 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.

`root` The root or top-level folder depending on your access level. All paths returned are relative to this root level. Permitted values are either `dropbox` or `app_folder`.

`revision` A **deprecated** field that semi-uniquely identifies a file. Use `rev` instead.

Note: `modified`, `rev`, and `revision` aren't returned in the metadata for the root/top-level path.

Errors

304 The folder contents have not changed (relies on `hash` parameter).
406 There are too many file entries to return.

/delta

Python
Java
Ruby
PHP

Description

A way of letting you keep up with changes to files and folders in a user's Dropbox. You can periodically call `/delta` to get a list of "delta entries", which are instructions on how to update your local state to match the server's state.

URL Structure

```
https://api.dropbox.com/1/delta
```

Method

POST

Parameters

cursor A string that is used to keep track of your current state. On the next call pass in this value to return delta entries that have been recorded since the cursor was returned.

locale The metadata returned will have its `size` field translated based on the given locale. For more information see above.

path_prefix If present, this parameter filters the response to only include entries at or under the specified path. For example, a `path_prefix` of "/Photos/Vacation" will return entries for the path "/Photos/Vacation" and any files and folders under that path. If you use the `path_prefix` parameter, you must continue to pass the same prefix on subsequent calls using the returned cursor.

include_media_info If true, each file will include a `photo_info` dictionary for photos and a `video_info` dictionary for videos with additional media info. When `include_media_info` is specified, files will only appear in delta responses when the media info is ready. If you use the `include_media_info` parameter, you must continue to pass the same value on subsequent calls using the returned cursor.

Returns

A JSON object with four fields:

entries A list of "delta entries" (described below).

reset If `true`, clear your local state before processing the delta entries. `reset` is always `true` on the initial call to `/delta` (i.e. when no cursor is passed in). Otherwise, it is `true` in rare situations, such as after server or account maintenance, or if a user deletes their app folder.

cursor A string that encodes the latest information that has been returned. On the next call to `/delta`, pass in this value.

has_more If `true`, then there are more entries available; you can call `/delta` again immediately to retrieve those entries. If 'false', then wait for at least five minutes (preferably longer) before checking again.

Delta Entries

Each delta entry is a 2-item list of one of the following forms:

`[<path>, <metadata>]` - Indicates that there is a file/folder at the given path. You should add the entry to your local state. The metadata value is the same as what would be returned by the `/metadata` call, except folder metadata doesn't have `hash` or `contents` fields. To correctly process delta entries:

If the new entry includes parent folders that don't yet exist in your local state, create those parent folders in your local state.

If the new entry is a file, replace whatever your local state has at path with the new entry.

If the new entry is a folder, check what your local state has at `<path>`. If it's a file, replace it with the new entry. If it's a folder, apply the new `<metadata>` to the folder, but don't modify the folder's children. If your local state doesn't yet include this path, create it as a folder.

`[<path>, null]` - Indicates that there is no file/folder at the given path. To update your local state to match, anything at path and all its children should be deleted. Deleting a folder in your Dropbox will sometimes send down a single deleted entry for that folder, and sometimes separate entries for the folder and all child paths. If your local state doesn't have anything at path, ignore this entry.

Note: Dropbox treats file names in a case-insensitive but case-preserving way. To facilitate this, the `<path>` values above are lower-cased versions of the actual path. The `<metadata>` value has the original case-preserved path.

/longpoll_delta

Description

A long-poll endpoint to wait for changes on an account. In conjunction with /delta, this call gives you a low-latency way to monitor an account for file changes.

Note that this call goes to **api-notify.dropbox.com** instead of api.dropbox.com.

Unlike most other API endpoints, this call does not require OAuth authentication. The passed in `cursor` can only be acquired via an authenticated call to /delta.

URL Structure

```
https://api-notify.dropbox.com/1/longpoll_delta
```

Method

GET

Parameters

cursor A delta cursor as returned from a call to /delta. Note that a cursor returned from a call to /delta with `include_media_info=true` is incompatible with /longpoll_delta and an error will be returned.

timeout An optional integer indicating a timeout, in seconds. The default value is 30 seconds, which is also the minimum allowed value. The maximum is 480 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.

Returns

The connection will block until there are changes available or a timeout occurs. In both cases, the HTTP status code will be 200, and the response will be a JSON dictionary. The value of the `changes` field indicates whether new changes are available. If this value is true, you should call /delta to retrieve the changes. If this value is false, it means the call to `/longpoll_delta` timed out.

If present, the value of the `backoff` field indicates how many seconds your code should wait before calling `/longpoll_delta` again.

Sample JSON return value

```
{"changes": false, "backoff": 60}
```

Errors

400 One or more parameters were invalid. The response will be of the form `{"error": "<reason>"}`.

/revisions

Python
Java
Ruby
PHP

Description

Obtains metadata for the previous revisions of a file.

Only revisions up to thirty days old are available (or more if the Dropbox user has Packrat). You can use the revision number in conjunction with the `/restore` call to revert the file to its previous state.

URL Structure

```
https://api.dropbox.com/1/revisions/<root>/<path>
```

root The root relative to which `path` is specified. Valid values are `sandbox` and `dropbox`.

path The path to the file.

Method

GET

Parameters

rev_limit Default is 10. Max is 1,000. Up to this number of recent revisions will be returned.

locale The metadata returned will have its `size` field translated based on the given locale. For more information see above.

Returns

A list of revisions formatted just like file metadata. More information on the returned metadata fields are available here.

Sample JSON return value

```
[
    {
        "is_deleted": true,
        "revision": 4,
        "rev": "40000000d",
        "thumb_exists": false,
        "bytes": 0,
        "modified": "Wed, 20 Jul 2011 22:41:09 +0000",
        "path": "/hi2",
        "is_dir": false,
        "icon": "page_white",
        "root": "app_folder",
        "mime_type": "application/octet-stream",
        "size": "0 bytes"
    },
    {
        "revision": 1,
        "rev": "10000000d",
        "thumb_exists": false,
        "bytes": 3,
        "modified": "Wed, 20 Jul 2011 22:40:43 +0000",
        "path": "/hi2",
        "is_dir": false,
        "icon": "page_white",
        "root": "app_folder",
        "mime_type": "application/octet-stream",
        "size": "3 bytes"
    }
]
```


/restore

Python
Java
Ruby
PHP

Description

Restores a file path to a previous revision.

Unlike downloading a file at a given revision and then re-uploading it, this call is atomic. It also saves a bunch of bandwidth.

URL Structure

```
https://api.dropbox.com/1/restore/<root>/<path>
```

root The root relative to which `path` is specified. Valid values are `sandbox` and `dropbox`.

path The path to the file.

Method

POST

Parameters

rev *required* The revision of the file to restore.

locale The metadata returned will have its `size` field translated based on the given locale. For more information see above.

Returns

The metadata of the restored file. More information on the returned metadata fields are available here.

Sample JSON response

```
{
    "is_deleted": true,
    "revision": 4,
    "rev": "40000000d",
    "thumb_exists": false,
    "bytes": 0,
    "modified": "Wed, 20 Jul 2011 22:41:09 +0000",
    "path": "/hi2",
    "is_dir": false,
    "icon": "page_white",
    "root": "sandbox",
    "mime_type": "application/octet-stream",
    "size": "0 bytes"
}
```

Errors

404Unable to find the revision at that path

/search

Python
Java
Ruby
PHP

Description

Returns metadata for all files and folders whose filename contains the given search string as a substring.

Searches are limited to the folder path and its sub-folder hierarchy provided in the call.

URL Structure

```
https://api.dropbox.com/1/search/<root>/<path>
```

root The root relative to which `path` is specified. Valid values are `sandbox` and `dropbox`.

path The path to the folder you want to search from.

Methods

GET, POST

Parameters

query *required* The search string. This string is split (on spaces) into individual words. Files and folders will be returned if they contain all words in the search string.

file_limit The maximum and default value is 1,000. No more than `file_limit` search results will be returned.

include_deleted If this parameter is set to true, then files and folders that have been deleted will also be included in the search.

locale The metadata returned will have its `size` field translated based on the given locale. For more information see above.

Returns

List of metadata entries for any matching files and folders. More information on the returned metadata fields are available here.

Sample JSON return value for a search for `.txt` in Dropbox/Public folder

```
[
    {
        "size": "0 bytes",
        "rev": "35c1f029684fe",
        "thumb_exists": false,
        "bytes": 0,
        "modified": "Mon, 18 Jul 2011 20:13:43 +0000",
        "path": "/Public/latest.txt",
        "is_dir": false,
        "icon": "page_white_text",
        "root": "dropbox",
        "mime_type": "text/plain",
        "revision": 220191
    }
]
```

/shares

Python
Java
Ruby
PHP

Description

Creates and returns a Dropbox link to files or folders users can use to view a preview of the file in a web browser.

Note: Links created after April 23rd, 2012 no longer expire after thirty days.

URL Structure

```
https://api.dropbox.com/1/shares/<root>/<path>
```

root The root relative to which `path` is specified. Valid values are `sandbox` and `dropbox`.

path The path to the file or folder you want to link to.

Method

POST

Parameters

locale Use to specify language settings for user error messages and other language specific text. See the notes above for more information about supported locales.

short_url When `true` (default), the url returned will be shortened using the Dropbox url shortener. If `false`, the url will link directly to the file's preview page.

Returns

A Dropbox link to the given path. The link can be used publicly and directs to a preview page of the file. For compatibility reasons, it returns the link's expiration date in Dropbox's usual date format. All links are currently set to expire far enough in the future so that expiration is effectively not an issue.

Sample JSON return value for a file

```
{
    "url": "https://db.tt/c0mFuu1Y",
    "expires": "Tue, 01 Jan 2030 00:00:00 +0000"
}
```

/media

Python
Java
Ruby
PHP

Description

Returns a link directly to a file.

Similar to `/shares`. The difference is that this bypasses the Dropbox webserver, used to provide a preview of the file, so that you can effectively stream the contents of your media.

URL Structure

```
https://api.dropbox.com/1/media/<root>/<path>
```

root The root relative to which `path` is specified. Valid values are `sandbox` and `dropbox`.

path The path to the media file you want a direct link to.

Method

POST

Parameters

locale Use to specify language settings for user error messages and other language specific text. See the notes above for more information about supported locales.

Returns

A url that serves the media directly. Also returns the link's expiration date in Dropbox's usual date format.

Sample JSON return value for a file

```
{
    'url': 'https://dl.dropboxusercontent.com/1/view/abcdefghijk/example',
    'expires': 'Fri, 16 Sep 2011 01:01:25 +0000'
}
```

Notes
The `/media` link expires after four hours, allotting enough time to stream files, but not enough to leave a connection open indefinitely.

/copy_ref
Python
Java
Ruby
PHP

Description
Creates and returns a copy_ref to a file. This reference string can be used to copy that file to another user's Dropbox by passing it in as the `from_copy_ref` parameter on /fileops/copy.

URL Structure

```
https://api.dropbox.com/1/copy_ref/<root>/<path>
```

root The root relative to which `path` is specified. Valid values are `sandbox` and `dropbox`.

path The path to the file you want a copy_ref to refer to.

Method
GET
Returns
A copy_ref to the specified file. For compatibility reasons, it returns the link's expiration date in Dropbox's usual date format. All links are currently set to expire far enough in the future so that expiration is effectively not an issue.

Sample JSON return value for a file

```
{
    "copy_ref": "z1X6ATl6aWtzOGq0c3g5Ng",
    "expires": "Fri, 31 Jan 2042 21:01:05 +0000"
}
```

/thumbnails
Python
Java
Ruby
PHP

Description
Gets a thumbnail for an image. Note that this call goes to **api-content.dropbox.com** instead of api.dropbox.com.

URL Structure

```
https://api-content.dropbox.com/1/thumbnails/<root>/<path>
```

root The root relative to which `path` is specified. Valid values are `sandbox` and `dropbox`.

path The path to the image file you want to thumbnail.

Method

GET

Parameters

format `jpeg` (default) or `png`. For images that are photos, `jpeg` should be preferred, while `png` is better for screenshots and digital art.

size One of the following values (default: `s`):

value	dimensions (px)
`xs`	32x32
`s`	64x64
`m`	128x128
`l`	640x480
`xl`	1024x768

Returns

A thumbnail of the specified image's contents. The image returned may be larger or smaller than the size requested, depending on the size and aspect ratio of the original image.

The HTTP response contains the content metadata in JSON format within an `x-dropbox-metadata` header.

Errors

404 The file path wasn't found or the file extension doesn't allow conversion to a thumbnail.
415 The image is invalid and cannot be converted to a thumbnail.

Notes

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.

/chunked_upload

Python
Java
Ruby
PHP

Description

Uploads large files to Dropbox in multiple chunks. Also has the ability to resume if the upload is interrupted. This allows for uploads larger than the `/files` and `/files_put` maximum of 150 MB.

Typical usage:

1. Send a PUT request to `/chunked_upload` with the first chunk of the file without setting upload_id, and receive an upload_id in return.
2. Repeatedly PUT subsequent chunks using the upload_id to identify the upload in progress and an offset representing the number of bytes transferred so far.
3. After each chunk has been uploaded, the server returns a new offset representing the total amount transferred.
4. After the last chunk, POST to `/commit_chunked_upload` to complete the upload.

Chunks can be any size up to 150 MB. A typical chunk is 4 MB. Using large chunks will mean fewer calls to `/chunked_upload` and faster overall throughput. However, whenever a transfer is interrupted, you will have to resume at the beginning of the last chunk, so it is often safer to use smaller chunks.

If the offset you submit does not match the expected offset on the server, the server will ignore the request and respond with a 400 error that includes the current offset. To resume upload, seek to the correct offset (in bytes) within the file and then resume uploading from that point.

A chunked upload can take a maximum of 24 hours before expiring.

URL Structure

```
https://api-content.dropbox.com/1/chunked_upload?param=val
```

param=val The URL-encoded parameters for this request. They cannot be sent in the request body.

Method

PUT

Request body

required A chunk of data from the file being uploaded. If resuming, the chunk should begin at the number of bytes into the file that equals the **offset**.

Parameters

upload_id The unique ID of the in-progress upload on the server. If left blank, the server will create a new upload session.

offset The byte offset of this chunk, relative to the beginning of the full file. The server will verify that this matches the offset it expects. If it does not, the server will return an error with the expected offset.

Returns

Sample JSON response

```
{
    "upload_id": "v0k84B0AT9fYkfMUp0sBTA",
    "offset": 31337,
    "expires": "Tue, 19 Jul 2011 21:55:38 +0000"
}
```

Errors

404 The `upload id` does not exist or has expired.

400 The `offset` parameter does not match up with what the server expects. The body of the error response will be JSON similar to the above, indicating the correct `offset` to upload.

/commit_chunked_upload

Description

Completes an upload initiated by the `/chunked_upload` method. Saves a file uploaded via `/chunked_upload` to a user's Dropbox.

`/commit_chunked_upload` is similar to `/files_put`. The main difference is that while `/files_put` takes the file contents in the request body, `/commit_chunked_upload` takes a parameter `upload_id`, which is obtained when the file contents are uploaded via `/chunked_upload`.

Note that this call goes to **api-content.dropbox.com** instead of api.dropbox.com.

URL Structure

```
https://api-content.dropbox.com/1/commit_chunked_upload/<root>/<path>
```

root The root relative to which `path` is specified. Valid values are `sandbox` and `dropbox`.

path The full path to the file you want to write to. This parameter should *not* point to a folder.

Method
POST
Parameters

locale The metadata returned on successful upload will have its `size` field translated based on the given locale.

overwrite This value, either `true` (default) or `false`, determines what happens when there's already a file at the specified path. If `true`, the existing file will be overwritten by the new one. If `false`, the new file will be automatically renamed (for example, `test.txt` might be automatically renamed to `test (1).txt`). The new name can be obtained from the returned metadata.

parent_rev The revision of the file you're editing. If `parent_rev` matches the latest version of the file on the user's Dropbox, that file will be replaced. Otherwise, the new file will be automatically renamed (for example, `test.txt` might be automatically renamed to `test (conflicted copy).txt`). If you specify a revision that doesn't exist, the file won't save (error 400). Get the most recent `rev` by performing a call to /metadata.

upload_id Used to identify the chunked upload session you'd like to commit.

Returns
The metadata for the uploaded file. More information on the returned metadata fields are available here.

Sample JSON response

```
{
    "size": "225.4KB",
    "rev": "35e97029684fe",
    "thumb_exists": false,
    "bytes": 230783,
    "modified": "Tue, 19 Jul 2011 21:55:38 +0000",
    "path": "/Getting_Started.pdf",
    "is_dir": false,
    "icon": "page_white_acrobat",
    "root": "dropbox",
    "mime_type": "application/pdf",
    "revision": 220823
}
```

Errors

400 Returned if the request does not contain an upload_id or if there is no chunked upload matching the given upload_id.

File operations

The various `fileops` calls provide the standard file operations. Files and folders can be moved, copied, or deleted. Folders can be created.

/fileops/copy

Python
Java
Ruby
PHP

Description
Copies a file or folder to a new location.

URL Structure

```
https://api.dropbox.com/1/fileops/copy
```

Method
POST
Parameters

root *required* The root relative to which `from_path` and `to_path` are specified. Valid values are `sandbox` and `dropbox`.

from_path Specifies the file or folder to be copied from relative to `root`.

to_path *required* Specifies the destination path, *including the new name for the file or folder*, relative to `root`.

locale The metadata returned will have its `size` field translated based on the given locale. For more information see above.

from_copy_ref Specifies a copy_ref generated from a previous /copy_ref call. Must be used instead of the `from_path` parameter.

Returns
Metadata for the copy of the file or folder. More information on the returned metadata fields are available here.

Sample JSON response

```
{
    "size": "15 bytes",
    "rev": "1f0a503351f",
    "thumb_exists": false,
    "bytes": 15,
    "modified": "Wed, 10 Aug 2011 18:21:29 +0000",
    "path": "/test1.txt",
    "is_dir": false,
    "icon": "page_white_text",
    "root": "dropbox",
    "mime_type": "text/plain",
```

```
    "revision": 496342
}
```

Errors

403 An invalid copy operation was attempted (e.g. there is already a file at the given destination, or trying to copy a shared folder).
404 The source file wasn't found at the specified path.
406 Too many files would be involved in the operation for it to complete successfully. The limit is currently 10,000 files and folders.

/fileops/create_folder

Python
Java
Ruby
PHP

Description

Creates a folder.

URL Structure

```
https://api.dropbox.com/1/fileops/create_folder
```

Method

POST

Parameters

root *required* The root relative to which `path` is specified. Valid values are `sandbox` and `dropbox`.

path *required* The path to the new folder to create relative to `root`.

locale The metadata returned will have its `size` field translated based on the given locale. For more information see above.

Returns

Metadata for the new folder. More information on the returned metadata fields are available here.

Sample JSON response

```
{
    "size": "0 bytes",
    "rev": "1f477dd351f",
    "thumb_exists": false,
    "bytes": 0,
    "modified": "Wed, 10 Aug 2011 18:21:30 +0000",
    "path": "/new_folder",
    "is_dir": true,
    "icon": "folder",
    "root": "dropbox",
    "revision": 5023410
}
```

Errors

403 There is already a folder at the given destination.

/fileops/delete

Python
Java
Ruby
PHP

Description
Deletes a file or folder.

URL Structure

```
https://api.dropbox.com/1/fileops/delete
```

Method
POST
Parameters

root *required* The root relative to which `path` is specified. Valid values are `sandbox` and `dropbox`.

path *required* The path to the file or folder to be deleted.

locale The metadata returned will have its `size` field translated based on the given locale. For more information see above.

Returns
Metadata for the deleted file or folder. More information on the returned metadata fields are available here.

Sample JSON response

```
{
    "size": "0 bytes",
    "is_deleted": true,
    "bytes": 0,
    "thumb_exists": false,
    "rev": "1f33043551f",
    "modified": "Wed, 10 Aug 2011 18:21:30 +0000",
    "path": "/test .txt",
    "is_dir": false,
    "icon": "page_white_text",
    "root": "dropbox",
    "mime_type": "text/plain",
    "revision": 492341
}
```

Errors
404 No file was found at the specified path.
406 Too many files would be involved in the operation for it to complete successfully. The limit is currently 10,000 files and folders.

/fileops/move
Python
Java
Ruby
PHP

Description

Moves a file or folder to a new location.

URL Structure

```
https://api.dropbox.com/1/fileops/move
```

Method

POST

Parameters

root *required* The root relative to which `from_path` and `to_path` are specified. Valid values are `sandbox` and `dropbox`.

from_path *required* Specifies the file or folder to be moved from relative to `root`.

to_path *required* Specifies the destination path, *including the new name for the file or folder*, relative to `root`.

locale The metadata returned will have its `size` field translated based on the given locale. For more information see above.

Returns

Metadata for the moved file or folder. More information on the returned metadata fields are available here.

Sample JSON response

```
{
    "size": "15 bytes",
    "rev": "1e0a503351f",
    "thumb_exists": false,
    "bytes": 15,
    "modified": "Wed, 10 Aug 2011 18:21:29 +0000",
    "path": "/test2.txt",
    "is_dir": false,
    "icon": "page_white_text",
    "root": "dropbox",
    "mime_type": "text/plain",
    "revision": 496342
}
```

Errors

403 An invalid move operation was attempted (e.g. there is already a file at the given destination, or moving a shared folder into a shared folder).
404 The source file wasn't found at the specified path.
406 Too many files would be involved in the operation for it to complete successfully. The limit is currently 10,000 files and folders.