|  |
| --- |
| @ |
| BOX.COM REST Connector |
| v.0.0.2 |

|  |
| --- |
| lakshman veti  11/7/2017 |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Reasons for Changes** | **Name** |
| 0.0.1 | 08-11-2017 | Initial Release | Lakshman Veti |
| 0.0.2 | 09-11-2017 | Added Groups, Collaborations | Lakshman Veti |
| 0.0.3 | 13-11-2017 | User authentication added | Lakshman Veti |
|  | 14-11-2017 | Updated authentication to JWT | Lakshman Veti |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Setup 5](#_Toc498249225)

[Authentication 5](#_Toc498249226)

[Users 5](#_Toc498249227)

[Create User 6](#_Toc498249228)

[Get User Information 6](#_Toc498249229)

[Update User 6](#_Toc498249230)

[Delete User 7](#_Toc498249231)

[Files 8](#_Toc498249232)

[File Upload 8](#_Toc498249233)

[File Download URL 8](#_Toc498249234)

[Get File Information 8](#_Toc498249235)

[Update File Info 9](#_Toc498249236)

[Delete File 9](#_Toc498249237)

[File Copy 10](#_Toc498249238)

[File Lock 10](#_Toc498249239)

[File Unlock 11](#_Toc498249240)

[File Versions 11](#_Toc498249241)

[Folders 11](#_Toc498249242)

[Folder Information 12](#_Toc498249243)

[Folder Items 12](#_Toc498249244)

[Create Folder 13](#_Toc498249245)

[Update Folder 13](#_Toc498249246)

[Delete Folder 13](#_Toc498249247)

[Groups 14](#_Toc498249248)

[Create Group 14](#_Toc498249249)

[Update Group 14](#_Toc498249250)

[Delete Group 15](#_Toc498249251)

[Get Group Information 15](#_Toc498249252)

[Add Member to Group 15](#_Toc498249253)

[Update membership 16](#_Toc498249254)

[Get Group Membership 16](#_Toc498249255)

[Remove membership 17](#_Toc498249256)

[Collaborations 17](#_Toc498249257)

[Create Collaboration 17](#_Toc498249258)

[Get the Collaborations on a File/Folder 17](#_Toc498249259)

# Setup

Inorder to run the BOX REST Connector, follow the below instructions.

#### Environment: NODE

#### SVN: <https://build.cloudseed.io:18080/svn/appgen1/branches/Appgen/box_com_API>

1. Checkout the code from about SVN Repo
2. Assuming the node is already installed (othewise [install](https://nodejs.org/en/download/)), navigate to checkedout folder “box\_com\_API” app from the console run command “*npm install*”
3. After succussful installation of all dependencies, run “*node rest*”
4. Console prints on which IP/HOST and PORT the app is running, that would be the rootURL of the REST app.
5. By default app listens to 8081, in order to change port number, pass port number as argument followed by command, ex: *node rest 8080*

# Authentication

Box REST connector (BRC) uses the standard JWT authentication process, which proceeds as follows:

1. The application sends login request along with user's unique ID (userID) to BRC for further authentication.
2. After a successful login, the authentication service returns an access token in response.
3. The application can then use the access token with API requests to gain access to the user's resources by including the token in an authorization header named **x-access-token**.

#### Endpoint: login/:userId

*Method: GET*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| userId (with URL) | Unique ID of user |
| x-access-token (header) | Each request to API has to send the auth-token along with x-access-token header |

*Returns: JSON string*

# Users

Users represent an individual's account on Box.

## Create User

To create a new user, call the below URL with necessary details, upon successful request responds with success/error json

#### Endpoint: user/create

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| Name | Name of the user |
| Email | Email of the user |
| is\_platform\_access\_only | true/false (true allows users only access through only API) |

*Returns: JSON string*

## Get User Information

#### To get a user information call the below URL with given details, requesting information for only the fields you need with the `fields` second parameter can improve performance and reduce the size of the network.

#### request.

#### Endpoint: user/get/:userId/:fields

*Method: GET*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| userId | Unique ID of the user |
| fields | Selective fields to be returned;  ex: user/get/1234/name,login,email if the complete user details are required just pass ‘null’ for fields, ex: user/get/123456/null |

*Returns: JSON string*

## Update User

To update an existing user, call the below URL with necessary details, upon successful request responds with success/error json

#### Endpoint: user/update/:userId

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| userId (with url) | Uniquer Identifier for user |
| Name | Name of the user |
| Email |  |

*Returns: JSON string*

## Delete User

To delete a user call the below endpoint with required inputs.

If the user still has files in their account and the `force` parameter is not sent, an error is returned.

If users needs to be deleted even if they have files in their account pass ‘force=true’ as part of request body

#### Endpoint: user/delete/:userId

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| userId (with URL) | User’s unique Identifier |
| Force | Ture / false, to remove evn if user has files exist |

*Returns: JSON string*

# Files

#### File objects represent individual files in Box. They can be used to download a file's contents, upload new versions, and perform other common file operations (move, copy, delete, etc.).

## File Upload

#### The simplest way to upload a file to a folder is by calling file/upload endpoint

#### Endpoint: file/upload/:folderId

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| folderId (with URL) | Parent folder ID |
| File | File attachment (with multipart-form) |

*Returns: JSON string*

## File Download URL

#### The download URL of a file can be retrieved by calling file/ downloadUrl endpoint

#### Endpoint: file/ downloadUrl/:fileId

*Method: GET*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| fileId (with URL) | File unique ID |

*Returns: JSON string*

## Get File Information

#### Calling file/get/:fileId returns a snapshot of the file's info.

#### Requesting information for only the fields you need with the `fields` second URL parameter can improve performance and reduce the size of the network request.

#### Endpoint: file/get/:fileId/:fields

*Method: GET*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| fileId | Unique ID of the file |
| fields | Only get information about a few specific fields  ex: file/get/123456/size,owned\_by if the complete file details are required just pass ‘null’ for fields, ex: file/get/123456/null |

*Returns: JSON string*

## Update File Info

To update an existing file, call the below URL with necessary details, upon successful request responds with success/error json

#### Endpoint: file/update/:fileId

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| fileId (with url) | Uniquer Identifier for file |
| Name | Name of the user |
| Email |  |

*Returns: JSON string*

## Delete File

#### Calling the file/delete/:fileId endpoint will move the file to the user's trash. Passing additional param through request body {isPermanent=true} removes file permanently, and cannot be retrieved later.

#### Endpoint: file/delete/:fileId

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| fileId (with URL) | File’s unique Identifier |
| isPermanent | true / false  if ‘true’ removes item permenantly  if ‘false’ or not passed this param, item moves to trash and the same can be retrieved |

*Returns: JSON string*

## File Copy

#### A file can be copied to a new folder with the file/copy endpoint. An optional `name` parameter can also be passed to rename the file on copy. This can be used to avoid a name conflict when there is already an item with the same name in the target folder.

#### Endpoint: file/file/:fileId/:newParentId

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| fileId (with URL) | File unique ID |
| newParentId (with URL) | New parent folder ID to which file to be moved. |

*Returns: JSON string*

## File Lock

#### A file can be locked, which prevents other users from editing the file, by calling the file/lock endpoint, Optionally prevent other users from downloading the file, as well as set an expiration time for the lock.

#### Endpoint: file/lock/:fileId/: expiryTime

*Method: GET*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| fileId (with URL) | File unique ID |
| expiryTime (with URL) | Duration of file expiration |

*Returns: JSON string*

## File Unlock

#### A file can be unlocked by calling below endpoint

#### Endpoint: file/unLock/:fileId

*Method: GET*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| fileId (with URL) | File unique ID |

*Returns: JSON string*

## File Versions

#### Retrieve a list of previous versions of a file by calling below endpoint. Requesting information for only the fields you need with the `fields` as part second URL parameter can improve performance and reduce the size of the network request.

#### Endpoint: file/versions/:fileId/:fields

*Method: GET*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| fileId (with URL) | File unique ID |
| Fields (with URL) | Only get information about a few specific fields  ex: : file/version/123456/name,size,sha1  if the complete file details are required just pass ‘null’ for fields, ex: file/version/123456/null |

*Returns: JSON string*

# Folders

Folder objects represent a folder from a user's account. They can be used to iterate through a folder's contents, collaborate a folder with another user or group, and perform other common folder operations (move, copy, delete, etc.)

## Folder Information

#### Folder information can be retrieved by calling the below endpoint. Use the second parameter in the URL to specify the desired fields. Requesting information for only the fields you need can improve performance and reduce the size of the network request.

#### Endpoint: folder/get/:folderId/:fields

*Method: GET*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| folderId (with URL) | Folder unique ID |
| Fields (with URL) | Only get information about a few specific fields  ex: : folder/get/123456/ name,shared\_link,permissions, sync\_state  if the complete file details are required just pass ‘null’ for fields, ex: folder/get/123456/null |

*Returns: JSON string*

## Folder Items

#### Folder items can be retrieved by calling the below endpoint. Use the `fileds` body parameter to specify the desired fields and control the result set paging(limit, offset). Requesting information for only the fields you need can improve performance and reduce the size of the network request.

#### Endpoint: folder/get/:folderId

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| folderId (with URL) | Folder unique ID |
| Fields (with body) | Only get information about a few specific fields  ex: : {fields:’name,shared\_link,permissions, sync\_state’} |
| offset (with body) | Currentr requesting set |
| limit (with body) | Retuslts per request |

*Returns: JSON string*

## Create Folder

#### Create a new folder under selected parent folder by calling the below endpoint

#### Endpoint: folder/create/:parentFolderId/:newFolderName

*Method: GET*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| parentFolderId (with URL) | Parent Folder ID under which the new folder to be created |
| newFolderName (with URL) | Name of the new folder |

*Returns: JSON string*

## Update Folder

#### Updating a folder's information is done by calling the below endpoint. Use the request body parameters to specify the fields to update, along with their new values.

#### Endpoint: folder/update/:folderId

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| folderId (with URL) | Folder unique ID |
| Fields (with body) | Updating other infromation  ex: : { sync\_state: 'synced'} |

*Returns: JSON string*

## Delete Folder

#### Deleting a folder is done by calling the below endpoint. To remove folder forcibily along with it’s contents pass recursive:true parameter as part of request body.

#### Endpoint: folder/delete/:folderId

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| folderId (with URL) | Folder unique ID |
| recursive (with body) | Deletes the folder along with folder contents  ex: : { recursive: true} |

*Returns: JSON string*

# Groups

Groups contain a set of users, and can be used in place of individual users in some operations, such as collaborations.

## Create Group

#### To create a new group, call the below endpoint, additional parameter like ‘description’ can be sent as part of request body.

#### Endpoint: group/create/:groupName

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| groupName (with URL) | New name for the group |
| description (with body) | Description for the group  ex: : { description: ‘group desc’} |

*Returns: JSON string*

## Update Group

#### To change the properties of a group object, call the below endpoint, additional parameter like ‘description’ can be sent as part of request body.

#### Endpoint: group/update/:groupId

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| groupId (with URL) | Unique ID of the group |
| name (with body) | name for the group  ex: : { name: ‘test’} |

*Returns: JSON string*

## Delete Group

To delete a group, call the below endpoint.

#### Endpoint: group/delete/:groupId

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| groupId (with URL) | Unique ID of the group |

*Returns: JSON string*

## Get Group Information

To retrieve the information for a group, call the below endpoint.

#### Endpoint: group/get/:groupId

*Method: GET*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| groupId (with URL) | Unique ID of the group |

*Returns: JSON string*

## Add Member to Group

#### To add a user to a group, call the below endpoint.

#### Endpoint: group/addMember/:groupID/:userId

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| groupId (with URL) | Unique ID of the group |
| userId (with URL) | User ID which is to be associated with group |
| role (with body) | The role of the user in the group. Default is “MEMBER” option for “ADMIN” |

*Returns: JSON string*

## Update membership

#### To update a membership record, call the below endpoint with `options` being the properties to update as part of request body.

#### Endpoint: group/ updateMembership/:membershipId

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| membershipId (with URL) | Unique ID of the membership |
| role (with body) | The role of the user in the group. Default is “MEMBER” option for “ADMIN”, Promote a user to group admin pass param 'role'  Ex: {role: ‘ADMIN’} |

## Get Group Membership

To retrieve information about a specific membership record, which shows that a given user is in the group, call the below endpoint. Requesting information for only the fields you need with the `fields` as part of second parameter can improve performance and reduce the size of the network request.

#### Endpoint: group/ getMembership/: membershipId/:fields

*Method: GET*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| membershipId (with URL) | Unique ID of the membership |
| fields | Selective fields to be returned;  ex: group/getMembership/1234/user,created\_at if the complete details are required just pass ‘null’ for fields, ex: group/getMembership/123456/null |

*Returns: JSON string*

## Remove membership

To remove member from a group, call the below endpoint.

#### Endpoint: group/ removeMember/: membershipId

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| membershipId (with URL) | Unique ID of the membership |

*Returns: JSON string*

# Collaborations

Collaborations are used to share folders between users or groups. They also define what permissions a user has for a folder.

## Create Collaboration

A collaboration can be added for an existing user with*below endpoint****.*** The `role` parameter determines what permissions the collaborator will have on the folder. You can create a collaboration on a file by setting the `type` option to `'file'`*/’folder****’***.

#### Endpoint: /collaboration/create/:userId/:type/:itemId/:role

*Method: POST*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| userId (with URL) | Unique userId who is to be added to collaboration |
| type (with URL) | ‘file’ / ‘folder’ to which user is associated |
| itemId (with URL) | File / folder ID |
| Role | Role of an user in the specified collaboration as listed below;  editor, viewer, previewer, uploader, previewer uploader, viewer uploader, or co-owner |

*Returns: JSON string*

## Get the Collaborations on a File/Folder

All of the collaborations on a file/folder can be retrieved by calling below endpoint***.***

#### Endpoint: collaboration/get/:itemId/:type

*Method: GET*

*Params:*

|  |  |
| --- | --- |
| Params | Description |
| itemId (with URL) | folderId / fileId |
| type | Collaboration type file/folder |

*Returns: JSON string*