Developers Guide to Cloud Winterstore (Admin SDK)



Welcome

We are excited for you to join us as we make client’s files more secure than before…and we hope you are here to stay and build future applications with us **Cloud Winterstore.**

Chapter 1 – Introduction

We have put a lot of effort to make Cloud Winterstore very easy to use, both in the application and the documentation. Our motto being to secure files and secure them easily.

Storage for your application can be implemented in 2 ways.

1. The client can fetch the file they want independently from the Storage Server.
2. The client requests to the server, and the server retrieves the file for the client from the Storage Server.

Both these implementations can be applied using Cloud Winterstore maintaining the same level of security.

**An exception stands for implementing (1) for web applications as the request will be blocked by the browser CORS and CORBS policy.**

Unless you know a way around these, we recommend architecture (2) for web-based applications or sites.

Cloud Winterstore exists because the services out there do not offer access control for individual files. Hence it is important for you to know that **Cloud Winterstore files are stored with each file having its own access control.**

**There are 4 key access control measures**, that are set when a file is uploaded by both the client and administrator.

* Allow All Users to Read
* Allow All Users to Write
* Allow Users with Key to Read
* Allow Users with Key to Write

***Allow All Users to Read***

If this is true, all users will be able to read the file. Reading is downloading the file.

***Allow All Users to Write***

If this is true, all users will be able to alter the file. This could be changing the name or the access control itself.

***Allow Users with Key to Write***

If this is true, only users that have keys to that file can write the file.

***Allow Users with Key to Read***

If this is true, only users that have keys to that file can read the file.

The above are very important points to note before any developer gets started with Cloud Winterstore.

Chapter 2 – Overview

The whole thing is put across 5 entities

**The Developer**

You are the developer who owns a project. Which you could have multiple projects.

You have the highest privilege in the project

**The Contributor (Admin Account)**

You can see this one in the Accounts section. This is a person who can contribute to the project. They can load your project and make changes, but can delete the project or remove you as the owner.

**The Client**

This is your client who is able to download and upload files. We like to see this as the application that is installed on the end user device. A client has a unique identification which could be an email or username.

**They possess a token they can use to upload and download a file**; without it they can do anything.

**Integration**

You as an administrator you can create and delete clients. We figured out that clients can be arranged as groups, and you may want to delete a group of clients at once. This is why we created integrations, a client belongs to one…and can’t exist without one. This therefore means you can delete multiple clients by deleting their integration.

The above noted, we can move on to the next chapter.

Chapter 3 - Admin SDK

The intent of the Admin SDK is providing the functions of the Console via command line. This is because creating client accounts, integrations etc. is supposed to be done automatically in your backend by running some function

FUNCTION CreateClient(parameters)

//Create a client account

We start off by listing all possible functions the Admin SDK can perform, which are

|  |  |
| --- | --- |
| Category | Description |
| Filing |  |
|  | View files in a folder (Folder ID) |
|  | View files in a folder (Folder Path) |
|  | Upload a file |
|  | Download a file (File ID) |
|  | Download a file (File Path) |
|  | Delete a file |
|  | Rename Index Object |
|  | Give keys to a client |
|  | Remove keys from a client |
|  | Get access control for a file |
|  | Create a folder |
|  | Get clients with keys to a file |
|  | Get file size |
|  | Get file name given id |
|  | Get file key given name |
|  | Get file last modification date |
|  | Get file owner |
| Download / Deleted Stats | |
|  | Get n download stats order of last inserted |
|  | Get download stats for an individual file |
|  | Get n deleted files stats order of last inserted |
|  | Get files that belong to a client |
| Client Accounts |  |
|  | Add a client account |
|  | Delete a client account |
|  | Search for a client account |
|  | Barn a client account |
|  | Unbarn a client account |
|  | Edit a client account |
|  | Get client account's Access Token |
|  | Get client account's last login |
|  | Get client account's creation date |
|  | Get client accounts by integration |
|  |  |
| Integrations |  |
|  | Add an integration |
|  | View integrations |
|  | Edit integration |
|  | Delete integration |
|  | Get integration's project |
| Integration Statistics |  |
|  | Get integration statistics |

Communication is done using Rest API. All the features above can be inflicted by sending an HTTP request to the server.

**98% of the request take in a JSON body in the request, except the a few that require multipart form data** i.e., to trigger the function to delete a file we can send a request to the server that looks like this.

|  |
| --- |
| **Headers** |
| Token “some token” |
| **Body (JSON)** |
| {  “id” : “some id”  } |

To the address <https://cloudwinterstore.co.zw/api/delete/> , to which the server will respond with a result. “200” meaning that the request was successful.

It is important to know that all files have a unique 64-character long identification that uniquely identify them from the rest.

Now that we know this, we can look at the functions independently.

Chapter 4 – View Files in A Folder (Folder ID)

**Description:**

Allows us to view files in a certain folder given a folder identification.

**Requirements:**

You need to be authenticated and have administrative privileges over a project, otherwise your request will be denied.

**Request URL**: https://cloudwinterstore.co.zw/console/api/get-folder/

**Request Type:** JSON Body (POST)

|  |
| --- |
| **Headers** |
| None |
| **Body (JSON)** |
| {  "projectName" : <project\_name>",  "folderID" : <folder\_id>  } |

**Notes:**

The root folder does not have an id similar to its subfolders. You will have to submit “root” as the folder\_id to view the files in the root directory. A project name is a combination of your username and the project name

<username>.<project\_name> i.e.

{

"projectName" : "rbryanben.Demo-Project",

"folderID" : "root"

}

Chapter 5 – View Files in A Folder (Folder Path)

**Description:**

Allows us to view files in a certain folder using a folder path

**Requirements:**

You need to be authenticated and have administrative privileges over a project, otherwise your request will be denied.

**Request URL**: https://cloudwinterstore.co.zw/console/api/get-path/

**Request Type:** JSON Body (POST)

|  |
| --- |
| **Headers** |
| None |
| **Body (JSON)** |
| {  "projectName" : <project\_name>",  "path" : <path>  } |

**Notes:**

This request only works if you are authenticated. All paths start with root.

A project name is a combination of your username and the project name

<username>.<project\_name> i.e.

{

"projectName" : "rbryanben.Demo-Project",

"path" : "root/Media/Drama/"

}

Chapter 6 – Upload A File

**Description:**

Allows us to upload files into a certain folder.

**Requirements:**

You need to have administrative privileges over a project, otherwise your request will be denied.

**Request URL**: https://cloudwinterstore.co.zw/console/upload-file

**Request Type:** Multipart Form data (POST)

|  |
| --- |
| **Headers** |
| None |
| **Body (Form)** Key: Value |
| “file” : <the\_binary\_file>  “name” : <name\_to\_store\_file\_as>  “size” : <computed\_size\_of\_file\_as\_bytes>  “project” : <name\_of\_project\_to\_upload\_to>  “parent” : <id\_of\_parent\_to\_upload\_file\_to>  “allowAllUsersWrite” : <Boolean>  “allowAllUsersWrite” : <Boolean>  “allowKeyUsersRead” : <Boolean>  “allowKeyUsersWrite” : <Boolean> |

**Notes:**

This request only works if you are authenticated. Assign :<id\_of\_parent\_to\_upload\_file\_to> as root, if the file is being uploaded to the root index.

Chapter 7 – Download a File (File ID)

**Description:**

Downloads a file given the file identification

**Requirements:**

You need to have administrative privileges over a project, otherwise your request will be denied.

**Request URL**: https://cloudwinterstore.co.zw/console/get-file

**Request Type:** JSON Body (POST)

|  |
| --- |
| **Headers** |
| None |
| **Body (JSON)** |
| {  “id” : <file\_id>  } |

**Notes:**

This request only works if you are authenticated.

Chapter 8 – Download a File (File Path)

**Description:**

Downloads a file given a file path

**Requirements:**

You need to be authenticated and have administrative privileges over a project, otherwise your request will be denied.

**Request URL**: https://cloudwinterstore.co.zw/console/api /get-file/path/

**Request Type:** JSON Body (POST)

|  |
| --- |
| **Headers** |
| None |
| **Body (JSON)** |
| {  "projectName" : <project\_name>",  "path" : <path>  } |

**Notes:**

This request only works if you are authenticated. All paths start with root.

A project name is a combination of your username and the project name

<username>.<project\_name> i.e.

{

"projectName" : "rbryanben.Demo-Project",

"path" : "root/Media/Drama/video1.mp4"

}

Chapter 9 – File Information

**Description:**

Gets information on a file given the file id

**Requirements:**

You need to be authenticated and have administrative privileges over a project, otherwise your request will be denied.

**Request URL**: https://cloudwinterstore.co.zw/console/api/file-info/

**Request Type:** JSON Body (POST)

|  |
| --- |
| **Headers** |
| None |
| **Body (JSON)** |
| {  "id" : <file\_identification>  } |

**Notes:**

{

"id" : "22BH7LNQYODCASX53971J5HWDBAO4SVWDPRPMSCN550SUZD084BHF5JI5Z2IZCT9"

}

Chapter 10 – Delete File

**Description:**

Deletes a file given a file ID

**Requirements:**

You need to be authenticated and have administrative privileges over a project, otherwise your request will be denied.

**Request URL**: https://cloudwinterstore.co.zw/api/delete/

**Request Type:** JSON Body (POST)

|  |
| --- |
| **Headers** |
| None |
| **Body (JSON)** |
| {  "id" : <file\_identification>  } |

**Notes:**

{

"id" : "22BH7LNQYODCASX53971J5HWDBAO4SVWDPRPMSCN550SUZD084BHF5JI5Z2IZCT9"

}

Chapter 11 – Rename Index Object

**Description:**

Renames a file or a folder given the id and a name

**Requirements:**

You need to be authenticated and have administrative privileges over a project, otherwise your request will be denied.

**Request URL**: https://cloudwinterstore.co.zw/api/rename-object/

**Request Type:** JSON Body (POST)

|  |
| --- |
| **Headers** |
| None |
| **Body (JSON)** |
| {  "id" : <file\_identification>,  “name’ : <new\_name>  } |

**Notes:**

{

"id" : "22BH7LNQYODCASX53971J5HWDBAO4SVWDPRPMSCN550SUZD084BHF5JI5Z2IZCT9",

“name” : “MyVideo.mp4”

}

Chapter 12 – Give Key

**Description:**

Gives a key to a client given the file id and the client identifier

**Requirements:**

You need to be authenticated and have administrative privileges over a project, otherwise your request will be denied.

**Request URL**: https://cloudwinterstore.co.zw/api/give-key/

**Request Type:** JSON Body (POST)

|  |
| --- |
| **Headers** |
| None |
| **Body (JSON)** |
| {  "account" : <account>,  “file” : <file\_id >  } |

**Notes:**

{

"file" : "22BH7LNQYODCASX53971J5HWDBAO4SVWDPRPMSCN550SUZD084BHF5JI5Z2IZCT9",

“account” : “client@cloudwinterstore.co.zw”

}