TERBINE JAVASCRIPT LIBRARY

Last Revision:	March 19, 2020
Version:	1.0.1

TABLE OF CONTENTS

Contents

OVERVIEW

This guide documents all of the methods available in the Javascript Library to be used to interface with the Terbine Application Programming Interface (API). This can be used in conjunction with the Terbine API Overview document, where detailed information to relevant services are found. Knowledge of Javascript, Promises, Blobs, and JSON are assumed. The following reference will provide a description of each function, along with examples of resolving the response promise using Async/Await or .Then/.Catch. A Terbine Javascript Library Tutorial containing more in-depth examples for usage can be found following this reference.

GETTING STARTED

Before you begin, you will need to install the library to use in your Javascript code. There are two ways to install the library, as an *npm* module or within a script tag for use in the browser.

To incorporate the Javascript Library using a script tag, download the Terbine Javascript Library file from the tools page and include the following within the head tag on any HTML page requiring its use:

```
Script Tag
```

```
<script type="text/javascript" src="/path/to/JSLibrary.js"></script>;
```

To download and install using npm, enter the following into the command line interface and include the package into your project:

```
npm
```

```
npm install terbine-js-library
```

INITIALIZE THE LIBRARY

Once installed, the Javascript Library must be initialized with the base uniform resource identifier (URI) of the API to interface with. To create the Terbine Library instance, you will need to pass the base URI, "https://api.terbine.io", as a parameter to the new JSLibrary constructor. The library's functions will then be available to use within your code.

```
Initialize
```

```
var baseURI = "https://api.terbine.io";
var terbine = new JSLibrary(baseURI);
```


AUTHORIZATION

LOGIN

The Login function logs a user in to Terbine's System and provides the user's **token** (`loginResult.token`) to be used in future functions requiring authentication, as well as the **orgID** (`loginResult.orgId`) for use in workspace functions.

Parameters

- Email (string): Email address associated with user's Terbine.io account
- Password (string): Password associated with user's Terbine.io account

Returns

- Promise containing JSON login response

Async/Await

```
// Within async function
var loginResult = await terbine.login(email, password);
```

Promise

```
terbine.login(email, password)
      .then(function(response) {
            // Code that relies on response
      })
      .catch(function(error) {
            // Code that relies on error
      });
```

Sample Result

```
{
      "lastlogin":"2020-01-29T11:40:45Z",
      "username":"example@terbine.com",
      "displayname":"User Display Name",
      "token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJqZW5hbEB0ZXJiaW5lLmNvbSIsIm
      V4cCI6MTU4MDM1NjE1Nn0.JwFZQZv0j1UTnK-bDPsvhqwKFUTcxe--7x9mNhRDE0U",
      "userid":"aab864d2-74fc-44c1-b0d3-632829719929",
      "orgid":"6d94ff99-4b41-42d0-b3a4-d14d89da0838",
      "orgname":"Company Name",
      "orgType":3,
      "supervisor":false,
      "tokenlifeseconds":36300,
      "numberNotices":0,
// Truncated
```


SEARCH

SEARCH SERVICE

The Search function sends a query to search Terbine's available metadata instances and allows for customization of displayed results. The response provides a list of these instances that match the parameters. A metadata instance's **guid**(`searchResult[index].id`) which is used in the Add to Workspace and List functions can be retrieved from this response.

Parameters:

- Query (string): Query word or unique ID to search
- PageNumber (integer): The result's page number to display
- PageSize (integer): The number of results to display on the page
- Sort (string): Sort by type, "DATEADDED"/"ALPHA"
- Order (string): Order of items, "DESC"/"ASC"

Returns:

- Promise containing the list of search results in JSON format

Async/Await

```
// Within async function
var searchResult = await terbine.search(query, pageNumber, pageSize,
      sort, order);
```

Promise

```
terbine.search(query, pageNumber, pageSize, sort, order)
      .then(function(response) {
            // Code that relies on response
      })
      .catch(function(error) {
            // Code that relies on error
      });
```

Sample Result

```
[
      {
            "Metadata":null,
            "Type":2,
            "Source":"ingestion",
            "Id":"9450a6e6-e0d5-43bf-9bb3-9ec59e9b87dc",
            "orgId":"6d94ff88-4b41-41d0-b3a4-d14d89da0939",
            "ownerOrgId":null,
            "alphaSort":"Nitric Oxide / Liberty Avenue, Lisbon, Portugal /
```



```
            2015",
            "title":"Nitric Oxide / Liberty Avenue, Lisbon, Portugal /
            2015",
            "description":"Readings of nitric oxide were taken by an air
            monitoring station located along the roadside of Liberty Avenue
// Truncated
```


WORKSPACE

LIST WORKSPACE

The List Workspace function displays a list of metadata instances currently added to a user's workspace. This response also provides each instance's **workspaceID** (`workspaceListResult[index].id`) which is required for use in the Lock/Flag/Delete functions.

Parameters:

- Token (string): Token retrieved from Login response
- OrgID (string): Organization ID retrieved from Login response
- PageNumber (integer): The result's page number to display
- PageSize (integer): The number of results to display on the page
- Order (string): Order of items, "DESC"/"ASC"
- OrderBy(string): Order by type, "description"/"title" etc.

Returns:

- Promise containing the list of current workspace items in JSON format

Async/Await

```
// Within async function
var workspaceListResult = await terbine.listWorkspace(token, orgID,
pageNumber, pageSize, order, orderBy);
```

Promise

```
terbine.listWorkspace(token, orgID, pageNumber, pageSize, order, orderBy)
      .then(function(response) {
            // Code that relies on response
      })
      .catch(function(error) {
            // Code that relies on error
      });
```

Sample Result

```
[
      {
            "typeId":1,
            "metadataType":"Archival",
            "id":"e0c4e29c-5780-4db0-91fe-feca64bf0158",
            "orgId":"c8dbd0ae-15a4-45ae-b595-6b75aea54de4",
            "name":"Water Levels / Oslo, Norway / 2014",
            "description":"Readings of water levels are taken by a float
            gauge located in Oslo, Norway. This float gauge is located
            within the Inner
// Truncated
```


ADD TO WORKSPACE

The Add to Workspace function adds a chosen metadata instance to the user's workspace where it can be later flagged, locked in for download, or deleted.

Parameters:

- Token (string): Token retrieved from Login response
- GUID (string): Unique Identifier of a Metadata Instance retrieved from Search response or Terbine.io
- OrgID (string): Organization ID retrieved from Login response

Returns:

- Promise containing the Added to Workspace response in JSON format

Async/Await

```
// Within async function
var addWorkspaceResult = await terbine.addToWorkspace(token, guid, orgID);
```

Promise

```
terbine.addToWorkspace(token, guid, orgID)
      .then(function(response) {
            // Code that relies on response
      })
      .catch(function(error) {
            // Code that relies on error
      });
```

Sample Result

```
{
      "typeId":null,
      "metadataType":null,
      "Id":"b81a3d5c-a5fd-4b62-a459-2b3390baa282",
      "orgId":"c8dbd0ae-15a4-45ae-b595-6b75aea54de4",
      "name":"Nitric Oxide / Lyndon B Johnson Freeway, Dallas, Texas, United
      States / 2015",
      "description":"Workspace entry added for Nitric Oxide / Lyndon B
      Johnson Freeway, Dallas, Texas, United States / 2015",
      "metadataId":"3c48c005-c3e6-4bf2-a968-5f35c69e27ad",
      "datasetId":null,
      "uid":"de4fe9f8-d68a-4592-9fb3-ba2ff8e8f5f7",
      "type":1,
      "status":1,
//Truncated
```


LOCK IN WORKSPACE

The Lock in Workspace function locks a metadata instance found in the workspace and activates the dataset/stream for use in the user's dashboard.

Parameters:

- Token (string): Token retrieved from Login response
- OrgID (string): Organization ID retrieved from Login response
- WorkspaceID (string): Unique ID of item in workspace

Returns:

- Promise containing the result of locking the workspace item in JSON format

Async/Await

```
// Within async function
var lockResult = await terbine.lockToWorkspace(token, orgID, workspaceID);
```

Promise

```
terbine.lockToWorkspace(token, orgID, workspaceID)
      .then(function(response) {
            // Code that relies on response
      })
      .catch(function(error) {
            // Code that relies on error
      });
```

Sample Result

```
{
      "typeId":null,
      "metadataType":null,
      "id":"e0c4e29c-5780-4db0-91fe-feca64bf0158",
      "orgId":"c8dbd0ae-15a4-45ae-b595-6b75aea54de4",
      "name":null,
      "description":null,
      "metadataId":"08a12f66-27b5-47f5-99d6-32088ac92116",
      "datasetId":null,
      "uid":"de4fe9f8-d68a-4592-9fb3-ba2ff8e8f5f7",
      "type":1,
      "status":3,
// Truncated
```

FLAG IN WORKSPACE

The Flag in Workspace function flags a metadata instance found in the workspace to distinguish it from other added workspace items.

Parameters:

- Token (string): Token retrieved from Login response
- OrgID (string): Organization ID retrieved from Login response
- WorkspaceID (string): Unique ID of item in workspace

Returns:

- Promise containing the result of flagging the workspace item in JSON format

Async/Await

```
// Within async function
var flagResult = await terbine.flagWorkspace(token, orgID, workspaceID);
```

Promise

```
terbine.flagWorkspace(token, orgID, workspaceID)
      .then(function(response) {
            // Code that relies on response
      })
      .catch(function(error) {
            // Code that relies on error
      });
```

Sample Result

```
{
      "typeId":null,
      "metadataType":null,
      "id":"e0c4e29c-5780-4db0-91fe-feca64bf0158",
      "orgId":"c8dbd0ae-15a4-45ae-b595-6b75aea54de4",
      "name":null,
      "description":null,
      "metadataId":"08a12f66-27b5-47f5-99d6-32088ac92116",
      "datasetId":null,
      "uid":"de4fe9f8-d68a-4592-9fb3-ba2ff8e8f5f7",
      "type":1,
      "Status":2,
// Truncated
```

DELETE FROM WORKSPACE

The Delete from Workspace function removes a metadata instance located in the workspace.

Parameters:

- Token (string): Token retrieved from Login response
- OrgID (string): Organization ID retrieved from Login response
- WorkspaceID (string): Unique ID of item in workspace

Returns:

- Promise containing the result of deleting the workspace item in JSON format

Async/Await

```
// Within async function
var deleteWorkspaceResult = await terbine.deleteFromWorkspace(token, orgID,
workspaceID);
```

Promise

```
terbine.deleteFromWorkspace(token, orgID, workspaceID)
      .then(function(response) {
            // Code that relies on response
      })
      .catch(function(error) {
            // Code that relies on error
      });
```

Sample Result

```
{
      "code":"OK",
      "message":"Workspace with id e0c4e29c-5780-4db0-91fe-feca64bf0158
      deleted",
      "messageDate":"2020-01-31T20:11:55Z"
}
```


METADATA

METADATA BY ID

The Metadata function returns the metadata field values for a certain metadata instance. The response also provides the **datasetID** (`metadataResult.dataset[index].id`) which is required for listing and downloading multiple files associated with the metadata instance.

Parameters:

- Token (string): Token retrieved from Login response
- Guid (string): Unique Identifier of a Metadata Instance retrieved from Search response or Terbine.io

Returns:

- Promise containing the values of all metadata fields for a metadata instance in JSON format

Async/Await

```
// Within async function
var metadataResult = await terbine.metadata(token, guid);
```

Promise

```
terbine.metadata(token, guid)
      .then(function(response) {
            // Code that relies on response
      })
      .catch(function(error) {
            // Code that relies on error
      });
```

Sample Result

```
{
      "id":"60ae666d-380f-45e2-b651-8a84dbb71c06",
      "orgId":"6d94ff88-4b41-41d0-b3a4-d14d89da0939",
      "orgName":"TERBINE",
      "identifier":{
            "createUpdateInfo":{
                  "createUser":"aab864c1-74fc-44c1-b0d3-531828717729",
                  "createDate":"2020-01-30T09:51:41Z",
                  "updateUser":"496fa3e3-d7e4-4ee9-aa96-feb213994c7f",
                  "updateDate":"2020-01-30T09:55:28Z"
            },
            "id":null,
            "extId":null,
            "urn":null,
            "uri":null,
```



```
            "tid":null
      },
      "meta":{
            "createUpdateInfo":{
                  "createUser":"aab864c1-74fc-44c1-b0d3-531828717729",
                  "createDate":"2020-01-30T09:51:41Z",
                  "updateUser":"496fa3e3-d7e4-4ee9-aa96-feb213994c7f",
                  "updateDate":"2020-01-30T09:55:28Z"
            },
      "name":"Electricity Consumption / Aube, Grand-Est, France / 2014",
      "description":"Measurements of electricity consumption data by
      activity sector with geographical grid IRIS on
// Truncated
```


DATASET

LIST FILES FOR DATASET

The List function returns the list of all files within a dataset. This response also provides the **contentID** (`listResult[index].id`) of each file which is required in the Download Single File function, as well as the original **file name** (`listResult[index].originalName`).

Parameters:

- Token (string): Token retrieved from Login response
- DatasetID (string): Dataset ID retrieved from Metadata response

Returns:

- Promise containing the list of files within a dataset in JSON format

Async/Await

```
// Within async function
var listResult = await terbine.list(token, datasetID);
```

Promise

```
terbine.list(token, datasetID)
	.then(function(response) {
		// Code that relies on response
	})
	.catch(function(error) {
		// Code that relies on error
	});
```

Sample Result

```
[
	{
	"id":"56783dd8-c02e-48a9-b5fd-ed35a6264c4e",
	"metadataId":"60ae666d-380f-45e2-b651-8a84dbb71c06",
	"datasetId":"abe07a75-bfab-40af-9666-d591fb18fcf2",
	"externalId":"1ee8381a70764511b0516a3129f53db3",
	"organizationId":"6d94ff88-4b41-41d0-b3a4-d14d89da0939",
	"type":1,
	"status":1,
	"fileStoreLocation":"6d94ff88-4b41-41d0-b3a4-d14d89da0939/60ae666d
	-380f-45e2-b651-8a84dbb71c06/GHxSqS1R3F8633PWLsgdisXN.terbine.xlsx",
	"fileExt":"xlsx",
	"size":998946,
// Truncated
```


DOWNLOAD SINGLE FILE

The Download Single File function provides a BLOB object containing the raw data of a single file within a dataset.

Parameters:

- Token (string): Token retrieved from Login response
- ContentID (string): Content ID retrieved from List response

Returns:

- Promise containing Blob of downloaded single file

Async/Await

```
// Within async function
var downloadSingleResult = await terbine.downloadSingle(token, contentID);
```

Promise

```
terbine.downloadSingle(contentID)
      .then(function(response) {
            // Code that relies on response
      })
      .catch(function(error) {
            // Code that relies on error
      });
```

Sample Result

```
Blob {
      size: 998946,
      type: "application/octet-stream"
}
```

DOWNLOAD MULTIPLE FILES

The Download Multiple File function provides a BLOB object containing the raw data of multiple files within a zip file.

Parameters:

- Token (string): Token retrieved from Login response
- DatasetID (string): Dataset ID retrieved from Metadata response
- Max (integer): maximum number of files to download

Returns:

- Promise containing Blob of the zip file containing multiple downloaded files

Async/Await

```
// Within async function
var downloadMultipleResult = await terbine.downloadMultiple(token,
datasetID, max);
```

Promise

```
terbine.downloadMultiple(token, datasetID, max)
      .then(function(response) {
            // Code that relies on response
      })
      .catch(function(error) {
            // Code that relies on error
      });
```

Sample Result

```
Blob {
      size: 996645,
      type: "application/zip"
}
```


TERBINE JAVASCRIPT LIBRARY TUTORIAL

OVERVIEW

This document provides more in depth examples of usage of the methods used in the Terbine Javascript Library. For simplicity, the examples shown were used in conjunction with an HTML file that uses a form to retrieve user input elements. Additionally, the Async/Await method used to resolve the response promises is shown.

GETTING STARTED

To begin, the Javascript Library must be downloaded through npm or downloaded. In this example we have downloaded the Javascript Library and have included it in the <head> tag of an HTML file.

```
<script src="/path/javascriptLibrary.js" type="text/javascript"></script>
```

INITIALIZE THE LIBRARY

In our script, we start by initializing a new library instance using the Base URI "https://api.terbine.io" and assigning it to the "terbine" variable.

```
// Create Library Instance
var terbine = new JSLibrary("https://api.terbine.io");
```


AUTHORIZATION

LOGIN

The Login function is necessary for many of the API calls found in the Javascript Library, as they require authentication of the user via a **token** (`loginResult.token`). Logging in also provides the user's **orgID** (`loginResult.orgid`) which is used to access their workspace. The Login function must be called prior to these functions.

After downloading the Javascript Library and creating an interface for input, we begin by adding the login function to our script. In this example, a login async function is created and email and password elements are retrieved from the user's input from the HTML form. The Login function is called and the loginResult variable then awaits its completion to retrieve the response. The response received can then be used to assign global token and orgID variables to be used in other functions.

Login

```
// Global Variables
var token;
var orgID;

async function login() {
      // Get user input from HTML form
      var email = document.getElementById("email").value;
      var pw = document.getElementById("password").value;

      // Call Login Function
      var loginResult = await terbine.login(email,pw);
      console.log("Login: ", loginResult);

      // Assign variables required by other calls
      token = loginResult.token;
      orgID = loginResult.orgid;
};
```


SEARCH

SEARCH SERVICE

The Search function sends a query to search Terbine's available metadata instances and allows for customization of displayed results using parameters. The parameters allowed include the page number (page displayed), the page size (items per page), sort (DATEADDED to order by date or ALPHA to order by name) and order (DESC for descending or ASC for ascending). From the result, a user can retrieve a metadata instance's **guid**(`searchResult[index].id`) which is used in the Add to Workspace and Metadata functions.

To add a Search Service in our program, we first add a search async function and retrieve query text and parameters from the user's input in an HTML form. The searchResult variable calls the function and awaits for the search call to complete and retrieves the response, an array of the search results. A global searchResult variable can then be assigned for use in the functions that require a guid.

Search

```
// Global Variable
var searchResult

async function search() {
      // Get user input from HTML form
      var query = document.getElementById("search").value;
      var pageNum = document.getElementById("pageNum").value;
      var pageSize = document.getElementById("pageSize").value;
      var sort = document.getElementsById('sort').value;
      var order = document.getElementsById('order').value;

      // Call Search Function
      searchResult = await terbine.search(query, pageNum, pageSize,
      sort, order);
      console.log("Search: ", searchResult);
};
```


WORKSPACE

LIST WORKSPACE

The List Workspace function displays a list of metadata instances currently added to a user's workspace. This call requires prior authentication by obtaining the token and orgID from the Login function. Similar to the Search function, parameters to customize the display of the resulting list include page number (page displayed), page size (items per page), order (DESC for descending or ASC for ascending), and orderBy (by description, title, etc). The response will provide each of the listed instances **workspaceIDs** (`workspaceListResult[index].id`), which is required for use in the Lock/Flag/Delete functions.

To access the workspace items, we create a listWorkspace async function and retrieve parameters from the user's input in the HTML form. The List Workspace function is then called and the workspaceListResult variable awaits for the call to complete. The retrieved response is an array of the list items. A global workspaceListResult variable can then be assigned for use in the workspace functions that require it.

List Workspace

```
// Global Variables
var token;
var orgID;
var workspaceListResult;

// Login Function to retrieve token and orgID

async function listWorkspace() {
      // Get user input from HTML form
      var wPageNum = document.getElementById("wPageNum").value;
      var wPageSize = document.getElementById("wPageSize").value;
      var wOrder = document.getElementsById('wOrder').value;
      var wOrderBy = document.getElementsById('wOrderBy').value;

      // Call List Workspace Function
      workspaceListResult = await terbine.listWorkspace(token, orgID,
      wPageNum, wPageSize, wOrder, wOrderBy);
      console.log("Workspace List:", workspaceListResult);
};
```

ADD TO WORKSPACE

The Add to Workspace function adds a chosen metadata instance to the user's workspace where it can be later flagged, locked in for download, or deleted. Authentication is required to access the workspace, so the Login function is called prior to retrieve the token and orgID. The GUID parameter, a unique identifier for a metadata instance, can be retrieved from a search result item or from Terbine.io's search service.

In order to add an item, an addWorkspace async function is created. The searchResult index of the metadata instance to be added is retrieved from the user's input in the HTML form. It is then used to retrieve the guid from the id of the item at that index. The Add to Workspace function is called by the addWorkspaceResult variable, which awaits for the call to complete and retrieves the response. The result will have a status of "1" when added.

```
Add to Workspace
```

```
// Global Variables
var token;
var orgID;
var searchResult;

// Login Function to retrieve token and orgID

// Search Function to retrieve guid

async function addWorkspace() {
      // Get user input from HTML form to retrieve guid
      var index = document.getElementById("index").value;
      var guid = searchResult[index].id;

      // Call Add to Workspace Function
      var addWorkspaceResult = await terbine.addToWorkspace(token, guid,
      orgID);
      console.log("Add to Workspace: ", addWorkspaceResult);
};
```

LOCK IN WORKSPACE

The Lock in Workspace function locks a metadata instance found in the workspace and activates the dataset/stream for use in the user's dashboard. Authentication is required to access the workspace, so the Login function is called prior to retrieve the token and orgID. Listing of the workspace is also required to retrieve the item to lock's workspaceID.

To lock an item, we create a lock async function. The index of the metadata instance to be locked from the workspaceListResult is retrieved from the user's input in the HTML form. It is then used to retrieve the workspaceID from the id of the item at that index. The Lock in Workspace function is called by the lockResult variable, which awaits for the call to complete and retrieves the response. The result will have a status of "3" when locked.

```
Lock in Workspace
```

```
// Global Variables
var token;
var orgID;
```



```
var workspaceListResult

// Login Function to retrieve token and orgID

// List Workspace Function to retrieve workspaceID

async function lock() {
      // Get user input from HTML form to retrieve workspaceID
      var lockIndex = document.getElementById("lockIndex").value;
      var workspaceID = workspaceListResult[lockIndex].id;

      // Call Lock to Workspace Function
      var lockResult = await terbine.lockToWorkspace(token,
      orgID, workspaceID);
      console.log("Lock to Workspace: ", lockResult);
};
```

FLAG IN WORKSPACE

The Flag in Workspace function flags a metadata instance found in the workspace to distinguish it from other added workspace items. Authentication is required to access the workspace, so the Login function is called prior to retrieve the token and orgID. Listing of the workspace is also required to retrieve the item to flag's workspaceID.

In this example, a flag async function is created. The index of the metadata instance to be flagged from the workspaceListResult is retrieved from the user's input in the HTML form. It is then used to retrieve the workspaceID from the id of the item at that index. The Flag in Workspace function is called by the flagResult variable, which awaits for the call to complete and retrieves the response. The result will have a status of "2" when flagged.

```
Flag in Workspace
```

```
// Global Variables
var token;
var orgID;
var workspaceListResult

// Login Function to retrieve token and orgID

// List Workspace Function to retrieve workspaceID

async function flag() {
      // Get user input from HTML form to retrieve workspaceID
      var flagIndex = document.getElementById("flagIndex").value;
      var fWorkspaceID = workspaceListResult[flagIndex].id;

      // Call Flag in Workspace Function
      var flagResult = await terbine.flagWorkspace(token, orgID,
      fWorkspaceID);
      console.log("Flag in Workspace: ", flagResult);
};
```


DELETE FROM WORKSPACE

The Delete from Workspace function removes a metadata instance located in the workspace. Authentication is required to access the workspace, so the Login function is called prior to retrieve the token and orgID. Listing of the workspace is also required to retrieve the item to delete's workspaceID.

To delete an item, we begin by creating a delete async function. The index of the metadata instance to be deleted from the workspaceListResult is retrieved from the user's input in the HTML form. It is then used to retrieve the workspaceID from the id of the item at that index. The Delete in Workspace function is called by the deleteResult variable, which awaits for the call to complete and retrieves the response. The result will show a code of "OK" if deleted successfully.

```
Delete from Workspace
```

```
// Global Variables
var token;
var orgID;
var workspaceListResult

// Login Function to retrieve token and orgID

// List Workspace Function to retrieve workspaceID

async function delete() {
      // Get user input from HTML form to retrieve workspaceID
      var deleteIndex = document.getElementById("deleteIndex").value;
      var dWorkspaceID = workspaceListResult[deleteIndex].id;

      // Call Delete From Workspace Function
      var deleteWorkspaceResult = await terbine.deleteFromWorkspace(token,
      orgID, dWorkspaceID);
      console.log("Delete from Workspace: ", deleteWorkspaceResult);
};
```


METADATA

METADATA BY ID

The Metadata function returns the metadata field values for a certain metadata instance. Authentication is required and the Login Function must be called prior to retrieve the token. The response provides the **datasetID** (`metadataResult.dataset[index].id`) within a dataset array. This is required for listing and downloading multiple files associated with the metadata instance.

To get the metadata of a dataset, a metadata async function is created and the index of the chosen item in the searchResult is retrieved from the user's input from the HTML form. It is then used to retrieve the guid from the id of the item at that index. The metadataResult variable calls the Metadata function and awaits the call to complete. The result is saved as a global variable for it to be used in future functions that require a datasetID.

Metadata

```
// Global Variables
var token;
var searchResult;
var metadataResult;

// Login Function to retrieve token

// Search Function to retrieve guid

async function metadata() {
      // Get user input from HTML form to retrieve guid
      var index = document.getElementById("index").value;
      var guid = searchResult[index].id;

      // Call Metadata Function
      metadataResult = await terbine.metadata(token, guid);
      console.log("Metadata: ", metadataResult);
};
```


DATASET

LIST FILES FOR DATASET

The List function returns the list of all files within a dataset. Authentication is required and the Login Function must be called prior to retrieve the token. The datasetID retrieved from the Metadata function result is also used as a parameter. The List function response provides the **contentID** (`listResult[index].id`) of each file which is required in the Download Single File function, as well as the original **file name** (`listResult[index].originalName`).

After calling the Metadata function, we create a list async function and retrieve the index from the user's input from the HTML form of the chosen item in the metadata dataset array. It is then used to retrieve the guid from the id of the item at that index. The listResult variable calls the List function and awaits the call to complete. The result is saved as a global variable for it to be used in the download of single items.

List

```
// Global Variables
var token;
var metadataResult;

// Login Function to retrieve token

// Metadata Function to retrieve metadataResult

async function list() {
      // Get user input from HTML form to retrieve datasetID
      var datasetIndex = document.getElementById("listIndex").value;
      var datasetID = metadataResult.dataset[datasetIndex].id;

      // Call List Function
      listResult = await terbine.list(token, datasetID);
      console.log("List: ", listResult);
};
```

DOWNLOAD SINGLE FILE

The Download Single File function provides a BLOB object containing the raw data of a single file within a dataset. Authentication is required and the Login Function must be called prior to retrieve the token. The Metadata and List functions must also precede this function to retrieve the contentID parameter.

After calling the List function, we create a downloadSingle async function and retrieve the index of the chosen item in the listResult array from the user's input from the HTML form. It is then used to retrieve the contentID and fileName from the id of the item at that

index. The downloadSingle variable calls the Download Single function and awaits the call to complete. The result is returned as a blob. This example also shows how to download the blob locally to the user's system by creating an <a> element.

Download Single File

```
// Global Variables
var token;
var metadataResult;
var listResult;

// Login Function to retrieve token

// Metadata Function to retrieve metadataResult

// List Function to retrieve listResult

async function downloadSingleTest() {
      // Get user input from HTML form to get contentID and fileName
      var contentIndex = document.getElementById("contentIndex").value;
      var contentID = listResult[contentIndex].id;
      var fileName = listResult[contentIndex].originalName;

      // Call Download Single Function
      var downloadSingle = await terbine.downloadSingle(token, contentID);
      console.log("Download Single Test: ", downloadSingle);

      // Download Locally
      var a = document.createElement("a");
      var url = window.URL.createObjectURL(downloadSingle);

      document.body.appendChild(a);
      a.style = "display: none";
      a.href = url;
      a.download = fileName;
      a.click();
      window.URL.revokeObjectURL(url);
};
```

DOWNLOAD MULTIPLE FILES

The Download Multiple File function provides a BLOB object containing the raw data of multiple files within a zip folder. Authentication is required and the Login Function must be called prior to retrieve the token. The Metadata function must also precede this function to retrieve the datasetID parameter.

After calling the Metadata function, we create a downloadMultiple async function and retrieve the index from the user's input from the HTML form of the chosen item in the metadataResult dataset array. It is then used to retrieve the datasetID from the id of the item at that index. The downloadMultiple variable calls the Download Single function

and awaits the call to complete. The result is returned as a blob. This example also shows how to download the blob locally to the user's system by creating an <a> element.

Download Multiple Files

```
// Global Variables
var token;
var metadataResult;

// Login Function to retrieve token

// Metadata Function to retrieve metadataResult

async function downloadMultiple() {
      // Get user input from HTML form to retrieve datasetID
      var datasetIndex = document.getElementById("datasetIndex").value;
      var datasetID = metadataResult.dataset[datasetIndex].id;

      // Call Download Multiple Function
      var downloadMulti = await terbine.downloadMultiple(token, datasetID);
      console.log("Download Multiple Test: ", downloadMulti);

      // Download Locally
      var a = document.createElement("a");
      var url = window.URL.createObjectURL(downloadMultiple);

      document.body.appendChild(a);
      a.style = "display: none";
      a.href = url;
      a.click();
      window.URL.revokeObjectURL(url);
};
```


REVISIONS

Version	Date	Name	Description
1.0.1	2020/03/19	Jenal Sanchez	Initial Release

