(Digits,Latin,Greek)[Scale=1]Roboto [Scale=1]Roboto

BIMData provides you:

- BIMData Connect: to authenticate and manage your applications
- an API to request data from your IFC files
- a Viewer for your IFC file in the brower

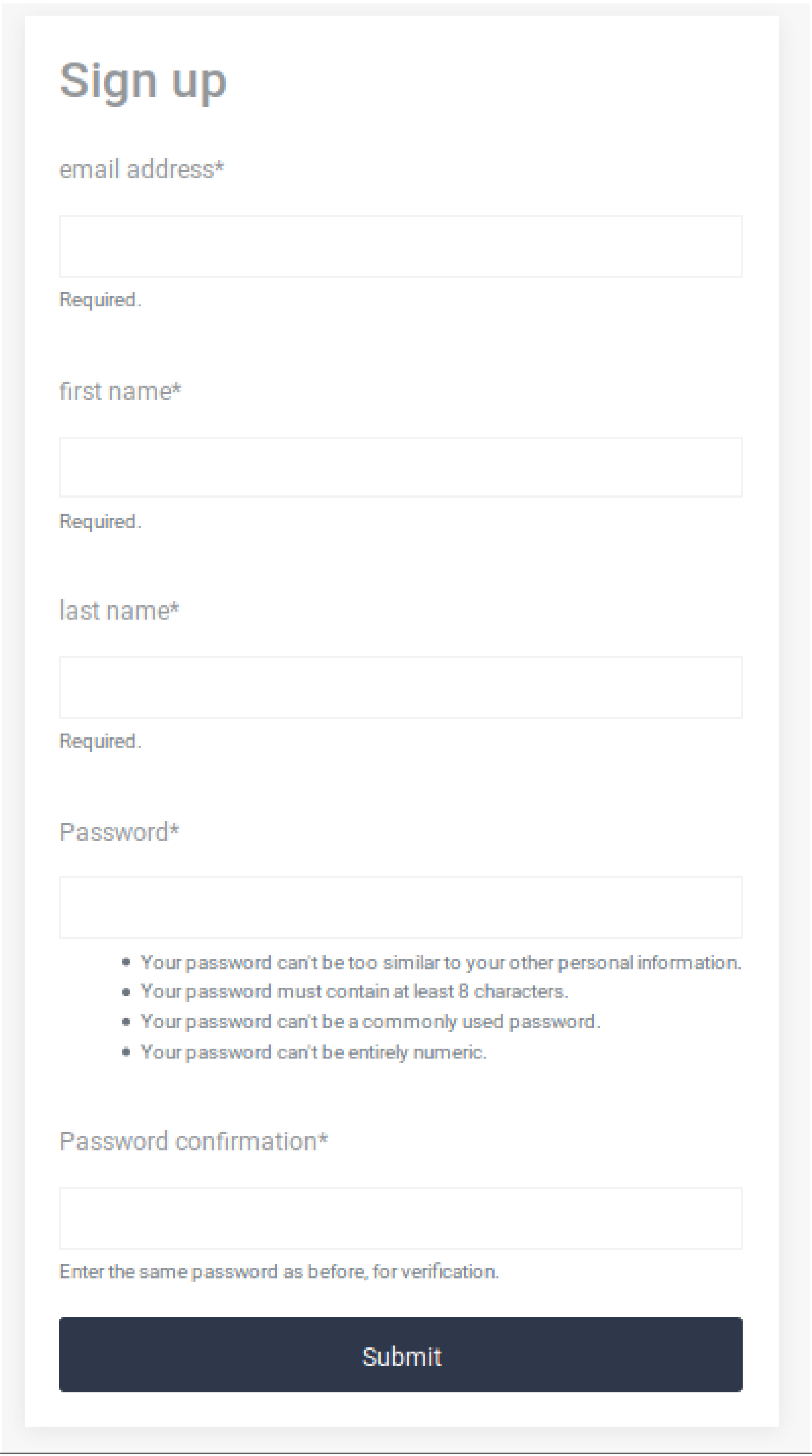
After reading about the concepts in our Concepts Tour, take a look at our Guide by topics, our Cookbook and our Tutorials.

1 Getting Started

1. Create a BIMData Connect account
2. Create your app
3. **Use the API**
    1. Use your credentials to log in
    2. Create your first cloud and its first project
    3. Upload your IFC file in your project
    4. Get all the information pieces from your IFC
4. BONUS: Include the Viewer on your website

1.1 Create a BIMData Connect account

First, you need an account. Fill the form with your e-mail and name, chose a password and you have your account.



Sign up

email address*

Required.

first name*

Required.

last name*

Required.

Password*

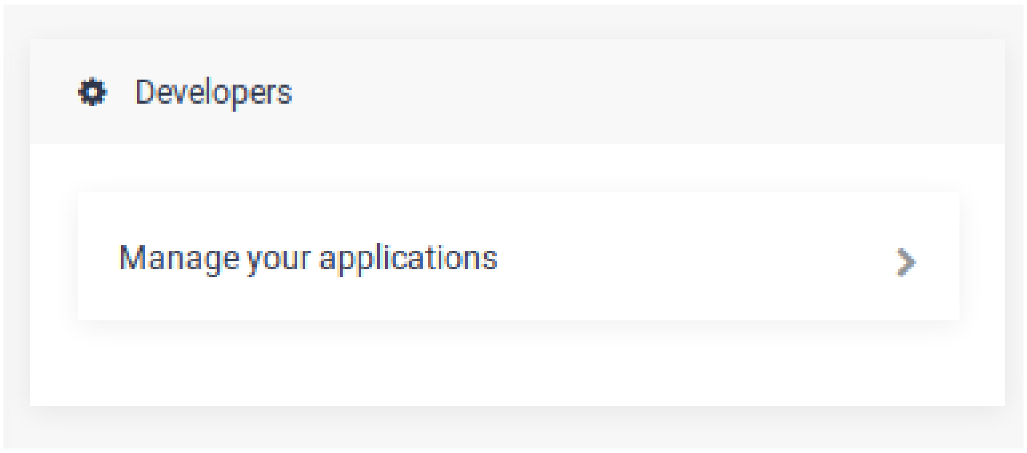
- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

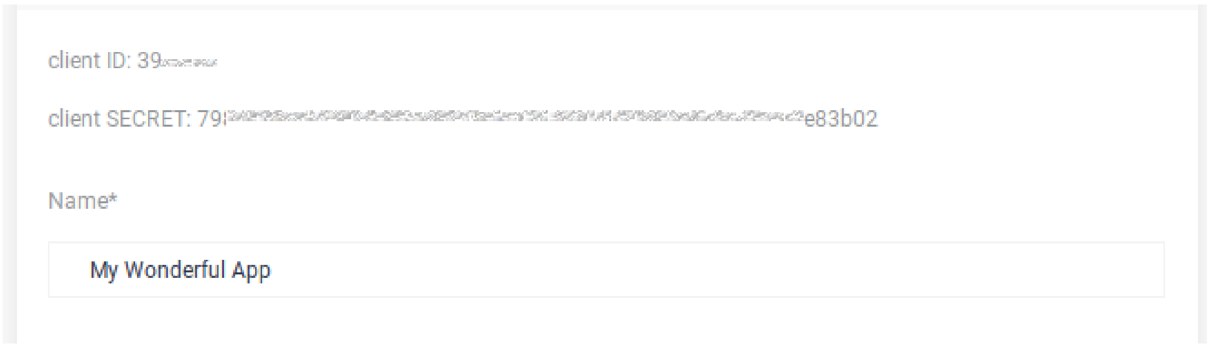
Enter the same password as before, for verification.

1.2 Create your app

In the BIMData Connect Account Manager screen, go to “Manage your applications” in the Developer area. There you can create an application.



Once your first app is created, you have a Client ID and a Client secret, this will help you with your API credentials.



client ID: 39[redacted]

client SECRET: 79[redacted]e83b02

Name*

My Wonderful App

1.3 Use the API

BIMData API is a tool to interact with your models stored on BIMData’s servers. Through the API, you can manage your projects, the clouds, upload your IFC files and manage them through endpoints.

1.3.1 Use your credentials to log in

The Client ID and the Client Secret are the 2 elements you need to get the Access Token of your app from the Identity Provider. You will need this Access Token for every call of the BIMData’s API.

1.3.2 Create your first cloud and its first project

See the details for the route /cloud/ for the cloud creation and for the route cloud/{cloud_pk}/project[1] the project creation.

```
response = requests.post(f'https://api-staging.bimdata.io/cloud', ␣
↪data=cloud_name, headers=headers)
cloud_id = response.json().get('id')
response = requests.post(f'https://api-staging.bimdata.io/cloud/{cloud_
↪id}/project',  headers=headers)
```

1.3.3 Upload your IFC file in your project

```
url = f'https://api-staging.bimdata.io/cloud/{cloud_id}/project/{project_
↪id}/document'
response = requests.post(url, data=payload, files=ifc_path_to_upload,␣
↪headers=headers)
```

Note: This is a step of our Viewer Tutorial

1.3.4 Get all the information pieces from your IFC

```
url = f'https://api-staging.bimdata.io/cloud/{cloud_id}/project/{project_
↪id}/document/{my_ifc_id}'
response = requests.get(url, data=my_filter, headers=headers)
```

Note: Many filters are available, check out the getElements route

1.4 Include the Viewer

See the dedicated page Getting Started with the Viewer

[1] https://developers-staging.bimdata.io/api/index.html#operation/createProject

2 Concepts

Cloud

A cloud is a global space where your projects are hosted.

Folders and documents

Folders and documents are useful to tidy your content.

IFC

After being uploaded, the IFC will be processed on our servers.

Projects

A Project is a place where IFC files and documents are stored.

Users management

Find out more about Users and BIMData Connect

Scopes

Using scopes is a way to handle the credentials of your application.

2.1 Cloud

2.1.1 Concept

A cloud is a set of *projects* sharing the same configuration. Each project contains your models, your Document Management System and BCFs.

Cloud administrators are also Projects admin by default, they can see every user in their cloud and change everyone's roles.

Cloud users can't see cloud collaborators. This means that a contractor on a project can't see every collaborator of the company.

2.1.2 References

- GET `/cloud`
- POST `/cloud`
- GET `/cloud/{cloud_pk}/user`
- GET `/cloud/{cloud_pk}/user/{user_pk}`
- GET `/cloud/{cloud_pk}/invitation`
- GET `/cloud/{cloud_pk}/size`
- GET `/cloud/{cloud_pk}/create-demo`

See also:

See also *3 - API Onboarding*

2.2 Folders & Documents

The API exposes a complete set of methods to upload and manage documents.

2.2.1 Folders

Every project is created with a root folder. It is the starting point to create a new folder or upload documents.

2.2.1.1 Code example

JSON

https://api-staging.bimdata.io/cloud/1/project/1

```
{
    "id": 1,
    "name": "my project",
    "cloud": {...},
    "status": "A",
    "created_at": "2017-12-01T10:09:54Z",
    "updated_at": "2018-02-21T17:07:25Z",
    "root_folder_id": 3,
}
```

- If a folder is created without parent_id, it will be placed under the root folder.
- You can't create a loop with folders (a parent being itself or a loop including multiple folders).

2.2.1.2 References

- GET `/cloud/{cloud_pk}/project/{project_pk}/folder`
- POST `/cloud/{cloud_pk}/project/{project_pk}/folder`
- GET `/cloud/{cloud_pk}/project/{id}/tree`

2.2.2 Documents

BIMData API allows you to upload any kind of file (IFC, Office, images, binaries, etc.). Those files are named `documents`. You can define in which folder you want to put the file using a `parent_id`.

2.2.2.1 Upload a document

File upload is one of the few API calls which does not use the `application/json` Content Type. This call uses `x-www-urlencoded` with `form-data`. The name of the file field must be "`file`", this means that you have to fire multiple calls if you want to upload many files.

Note: The filesize is the compressed size and not the actual size of the initial file due to HTTP Compression.

cURL

```
curl -X POST \
'https://api-staging.bimdata.io/cloud/1/project/1/document' \
-H 'authorization: Bearer ZeZr9oYxHspA8OdSCo9uftaLaEHX1N'
-H 'content-type: multipart/form-data; boundary=----
↪WebKitFormBoundary7MA4YWxkTrZu0gW' \
-F name=my_custom_name \
-F file=@/path/to/XXX.pdf
```

Python

```
import requests

url = "https://api-staging.bimdata.io/cloud/1/project/1/
↪document"

headers = {
    'authorization': 'Bearer ZeZr9oYxHspA8OdSCo9uftaLaEHX1N',
```

(continues on next page)

(continued from previous page)

```
}

payload = {
    'name': 'my_custom_name'
}

files = {'file': open('/path/to/XXX.pdf', 'rb')}

response = requests.request("POST", url, data=payload,␣
↪files=files, headers=headers)

print(response.text)
```

JavaScript

```
var fs = require("fs");
var request = require("request");

var options = { method: 'POST',
url: 'https://api-staging.bimdata.io/cloud/1/project/1/document
↪',
headers:
{ 'authorization': 'Bearer ZeZr9oYxHspA8OdSCo9uftaLaEHX1N',
    'content-type': 'multipart/form-data; boundary=----
↪WebKitFormBoundary7MA4YWxkTrZu0gW' },
formData:
{ name: 'my_custom_name',
    file:
    { value: 'fs.createReadStream("/path/to/XXX.pdf")',
        options: { filename: '/path/to/XXX.pdf', contentType:␣
↪null } } } };

request(options, function (error, response, body) {
if (error) throw new Error(error);

console.log(body);
});
```

2.2.2.2 Response

```
{
    "id": 424,
    "parent": 1,
    "creator": 134,
    "project": "1",
```

(continues on next page)

(continued from previous page)

```
    "name": "my_custom_name",
    "file_name": "XXX.pdf",
    "description": null,
    "file": "https://storage.gra3.cloud.ovh.net/v1/AUTH_
↪b6a1c0b6b7c041d3a71d56f84ce25102/bimdata-staging-dev/cloud_1/project_1/
↪XXX.pdf?temp_url_sig=311d34059bbebc87cd7f37de244bb6b62d114679&temp_url_
↪expires=1527771256",
    "size": 175780,
    "created_at": "2018-05-31T12:24:16Z",
    "updated_at": "2018-05-31T12:24:16Z",
    "ifc_id": null,
    "parent_id": 1
}
```

2.2.2.3 Download a document

You can download files using the URL returned by the API. The URL is valid for 1 hour.

cURL

```
curl -X GET \
'https://storage.gra3.cloud.ovh.net/v1/AUTH_
↪b6a1c0b6b7c041d3a71d56f84ce25102/bimdata-staging-dev/cloud_1/
↪project_1/XXX.pdf?temp_url_
↪sig=311d34059bbebc87cd7f37de244bb6b62d114679&temp_url_
↪expires=1527771256'
```

Python

```
import requests

url = "https://api-staging.bimdata.io/cloud/1/project/1/ifc"

querystring = {"status":"C"}

headers = {
    'Content-Type': "application/json",
    'Authorization': "Bearer ZeZr9oYxHspA8OdSCo9uftaLaEHX1N",
    }

response = requests.request("GET", url, headers=headers,␣
↪params=querystring)

print(response.text)
```

JavaScript

```
import requests

url = "https://storage.gra3.cloud.ovh.net/v1/AUTH_
↪b6a1c0b6b7c041d3a71d56f84ce25102/bimdata-staging-dev/cloud_1/
↪project_1/XXX.pdf?temp_url_
↪sig=311d34059bbebc87cd7f37de244bb6b62d114679&temp_url_
↪expires=1527771256"

response = requests.request("GET", url)

print(response.text)
```

2.2.2.4 References

- GET `/cloud/{cloud_pk}/project/{project_pk}/document`
- POST `/cloud/{cloud_pk}/project/{project_pk}/document`

See also:

See also *3 - API Onboarding*

2.3 IFC

BIMData API exposes a lot of tools for extract, update and manipulate information from IFC files[2].

The tools are compatible IFC2x3TC1 and IFC4 Add2.

Depending on the options you chose, you can:

- Retrieve the model as a 3D GLTF file
- Retrieve elements and properties
- Retrieve the spatial structure
- Retrieve classifications, systemes and zones
- Retrieve 2D plans in SVG format

2.3.1 Upload an IFC

To upload an IFC file, you have to upload a `document`. When the BIMData API detects an IFC format (based on the file name ending with `.ifc` or `.ifczip`), it will trigger the IFC process.

IFC files are tied to a `document` which represents the actual uploaded file.

We use HTTP Compression to speed up the file transfer. HTTP Compression will start as soon as you upload a file. Files are decompressed at the output of the API.

[2] https://en.wikipedia.org/wiki/Industry_Foundation_Classes

Note: The displayed filesize is the compressed size and not the actual size of the initial file.

2.3.2 Workflow

After being uploaded, the IFC will be processed on our servers.

Note: The process takes from few minutes to an hour depending on the size of the file and the options activated.

You can follow the progress on the `status` field:

status	Name of the status	Description
P	Pending	Your IFC will soon be processed
I	In process	The process has started
C	Completed	The process is complete and you can retrieve data from the API
E	Error	The process has failed. It's more likely to be a problem on our side. An alert is triggered and our team will fix it promptly.

See also:

See also *webhooks content*

Webhooks allow you to build automation around BIMData API.

2.4 Projects

2.4.1 Concept

A project is a place where IFC files and documents are stored. IFC files and documents can be uploaded and organized, checkplans are defined.

A project is attached to a cloud and a cloud can host an infinite number of projects.

A project contains:

- your models
- your Document Management System
- and BCFs.

Note: A BCF is linked to a project, not a model.

A project member can see all other members, and admin member can add a user to the project.

2.4.2 References

- GET `/user/projects`
- GET `/cloud/{cloud_pk}/project`
- POST `/cloud/{cloud_pk}/project`

See also:

See also *Getting Started* to learn how-to setup your project.

Follow the guide and make your first steps into the BIMData's API.

2.5 Users

2.5.1 Concept

Users are currently heavily linked with the BIMData.io Platform in the current version : https://login-staging.bimdata.io

A user can be in none or many clouds (many-to-many relation) with the role of User or Administrator. A user can be in none or many *Projects* (many-to-many relation) with the role of User or Administrator.

See also:

See also *authentication guide*.

2.6 Scopes

A scope is an important concept using the API. Using scopes is a way to handle the credentials of your application.

2.6.1 What's a scope?

A scope is a limitation to the data on a given resource. A scope is described by two words: the resource and the limitation, i.e. ifc:write Access Token is validated by the BIMData Connect authentication service and the scopes are attached to an Access Token.

Note: About scopes and oAuth 2.0 OAuth 2.0 scopes provide a way to limit the amount of access that is granted to an access token. For example, an access token issued to a client app may be granted READ and WRITE access to protected resources, or just READ access. You can implement your APIs to enforce any scope or combination of scopes you wish. So, if a client receives a token that has READ scope, and it tries to call an API endpoint that requires WRITE access, the call will fail.

source : https://docs.apigee.com/api-platform/security/oauth/working-scopes

Your application's user sees the scopes you registered as granted for your application and gives consent to the usage of their data based on this information. Set only the scopes you need.

The limitations are:

- Read: access to the data in read-only mode
- Write: edit the data
- Manage: link the elements, create/delete the links between elements

2.6.1.1 List of scopes available

- bcf:read, bcf:write
- check:read, check:write
- cloud:read, cloud:manage
- document:read, document:write
- ifc:read, ifc:write
- org:manage
- user:read
- webhook:manage

2.6.2 How to set the scopes of your application

The resources and possible scopes are pre-defined.

You can set a scope by typing scopes in a list in the form field Scopes. Each line contains only one scope. In the Manage your application screen, you can add, edit or remove from the Scopes list the granted access.

2.6.2.1 The scopes available

Table 1: Scopes in table format

Resource	Read	Write	Manage
bcf	x	x	
check	x	x	
cloud			x
document	x	x	
ifc	x	x	
org			x
user	x		
webhook			x

See also:

See also *security guide*.

3 Guide

API Introduction

Discover BIMData's API documentation

Authentication

BIMData Connect handles sign-in and logs in for your app.

Authentication migration

The new Authentification process is slightly different, learn how to migrate.

Errors

BIMData uses conventional HTTP response codes to indicate the success or failure of an API request.

Filters

Many API end-points allow filtering.

Webhooks

Webhooks allow you to build automation around BIMData API.

Security

Here are answers on how we keep your data secure.

3.1 BIMData's API

3.1.1 Introduction

BIMData.io let you access to your clouds, projects and all the data in your IFC files through the API. One your account on BIMData Connect is created, you can:

- Create and manage clouds
- Create and manage projects
- upload IFC file
- request data from clouds, projects, and models

3.1.2 API Reference

Just looking for the API Reference?

3.1.3 URL

The API URL is https://api-staging.bimdata.io

All API requests must be made over HTTPS.

Note: Calls made over plain HTTP will respond a 302, redirecting to the same URL over HTTPS.

3.1.3.1 API Enpoint

```
https://api-staging.bimdata.io
```

3.1.4 Our Tutorials

Read our tutorials to begin using the API with a real-life sized purpose

- Get data from model into an Excel file

  You want statistics about which Elements are the most commented with BCF.
- Retrieve elements following a constraint

  How-to retrieve elements of your model through the API

3.1.5 Playground

You would like to try our API? Gladly, we provide you some playground, based on OpenAPI files : check it out!

3.1.6 Libraries

We're currently maintaining two external libraries: * Our external lib in JavaScript[3] * Our external lib in Python[4].

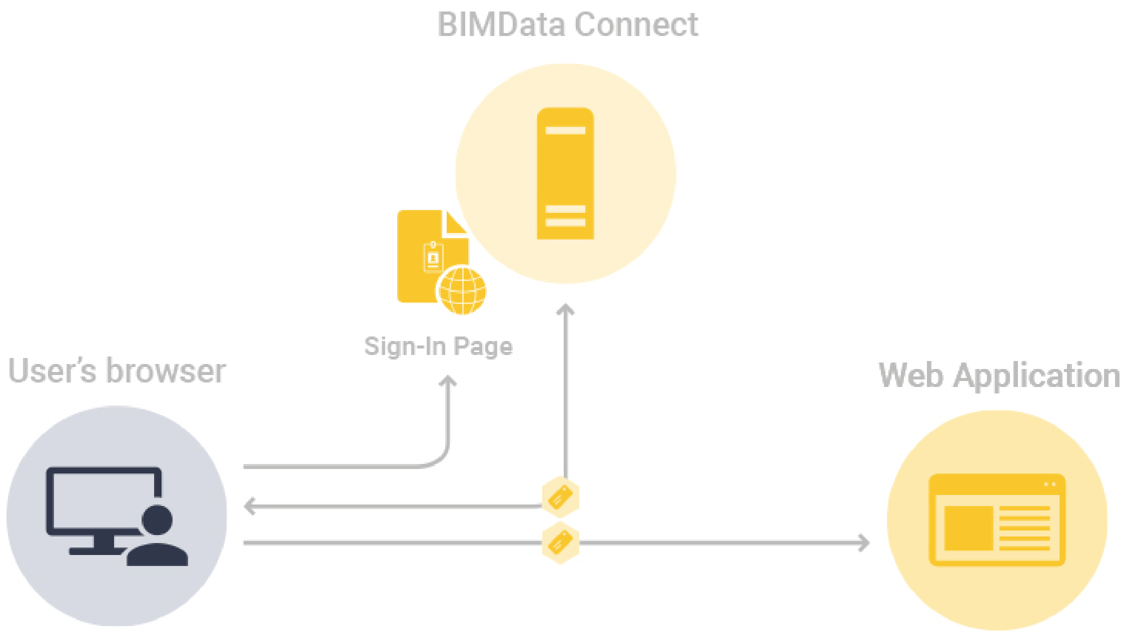
They are auto-generated from our OpenAPI file[5] with openapi-generator[6].

3.2 Authentication with BIMData Connect

The OpenID Connect used by the BIMData Connect, our authentication system, is built on the shoulders of OAuth2.0.

BIMData Connect handles the sign-in, the login and authentication processes of your application users. You can focus on creating and building your application. The user's browser is redirected to the Sign-In page by the Web Application.

The Sign-In page is on the BIMData Connect server. The BIMData Connect provides to the user's browser an Access Token. Then the user's browser could send requests to the Web Application sending the Access Token. The type of authentication is defined during the creation of the application.



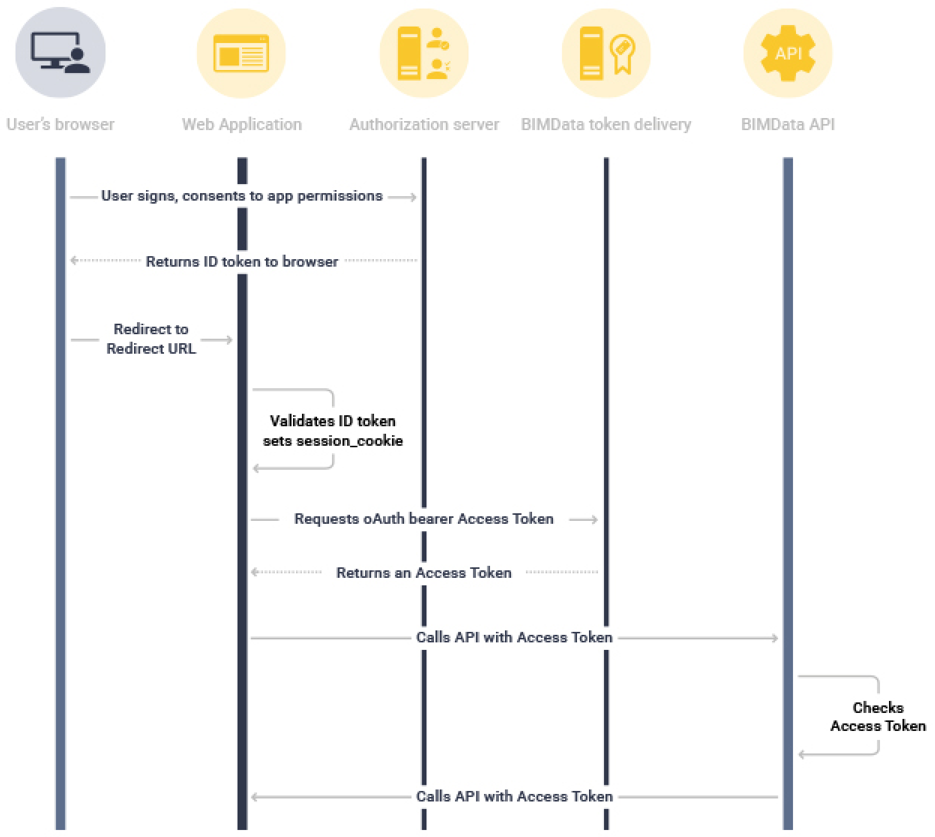
BIMData Connect handles sign-in and log in for your app

[3] https://www.npmjs.com/package/@bimdata/bimdata-api-client
[4] https://pypi.org/project/bimdata-api-client/
[5] https://api-beta.bimdata.io/doc#/
[6] https://github.com/OpenAPITools/openapi-generator

3.2.1 Get your Access Token

> **Warning:** Requirement: you must have an application, see Create an application process.

Follow the procedure described in Authentication by client credential


User's browser
Web Application
Authorization server
BIMData token delivery
BIMData API
API
User signs, consents to app permissions
Returns ID token to browser
Redirect to Redirect URL
Validates ID token sets session_cookie
Requests oAuth bearer Access Token
Returns an Access Token
Calls API with Access Token
Checks Access Token
Calls API with Access Token

Authentication flow

3.2.2 Use your Access Token

There are two possible ways to authenticate depending on your application architecture design. You can either:

- access as an application and benefit from authentication capacities,
- or use a user-behavior authentication.

3.2.3 When use an app auth?

3.2.3.1 The benefits

Simple to use

No user means no credentials to manage nor complex workflow, it's simpler to access via the application.

Pluggable

You can subscribe to events and use webhooks. It's the easiest way to provide automation.

Use it when you need to have a scheduled response to an event and launch a script depending on this response.

Important: You cannot access as a user, therefore you cannot: * do any impersonation * manage fine granularity with access rights * share data with other applications using BIMData

3.2.4 When use a user impersonation?

3.2.4.1 The benefits

User's name as the author

Emulating the user's actions enables you to act in the name of the user. Creating content with impersonation writes the user's name in the creator's name of this content.

Sharing the authoring

Your script can modify data created by the user and amend it.

Let BIMData handle the complexity

The credentials complexity is handled by the BIMData Connect authentication server. This option is compliant with the user's credentials.

Use it when you need to access the user's log, such as the user's history, and report actions.

There are three types of user auth, detailed beneath:

- Authorization code flow
- Implicit flow
- Hybrid flow

3.2.5 Type of user auth detailed

The three types are three different mechanisms to aks for user's permissions.

See also:

See also the tutorial *"Retrieve elements following a constraint"* and *all the recipes from our Cookbook*

3.3 Authentication migration

The authentication migration is currently under progress. Your old credentials are still working with the new Authentication, however the association of the users credentials with their clouds is no longer provided.

Warning: The previous authentication will no longer be available from 30th April 2019.

3.3.1 What's new?

- The new authentication process no longer need your username nor your password.
- URL is not the same than the previous one.
- Grant type is now: 'client_credentials'

The new authentication is fully implementing OpenID Connect offering to the users a more secure authentication. Additionally to that, respecting the OpenID Connect standard means more features such as authenticate users from your front or your back apps, emulate the user's actions and also create your own credentials.

Learn more about the new authentication on the dedicated page.

3.3.2 Comparison of Requests

Comparing the old and new (current) requests and responses shows the similarities between the old and new authentications.

3.3.2.1 Old Authentication Request

The old request was a string payload with username and password in it.

```
import requests
url = "https://api-staging.bimdata.io/oauth/v2/token/"
payload = "client_secret=CLIENT_SECRET&client_id=CLIENT_ID&grant_
↪type=password&username=my_user%40mail.com&password=passw0rd"
response = requests.post(url, data=payload, headers=headers)
```

3.3.2.2 New Authentication Request

The new request takes a JSON payload.

```
import requests
url = "https://login-staging.bimdata.io/token"
payload = {
```

(continues on next page)

(continued from previous page)

```
    "client_id": CLIENT_ID,
    "client_secret": CLIENT_SECRET,
    "grant_type": "client_credentials",
}

response = requests.post(url, data=payload)
```

3.3.3 Comparison of Responses

3.3.3.1 Old Response

The old response contains a “refresh_token” property.

```
{
    "access_token": "ZeZr9oYxHspA99dSCo9uftaLaEHX1N",
    "expires_in": 36000,
    "token_type": "Bearer",
    "scope": "read write",
    "refresh_token": "MI8fx999PmcfdEDYntvBIKck999BuM"
}
```

3.3.3.2 New Response

The new response has a more detailed scope.

```
{
    "access_token": "108a3781a1234a9d84a12345bd863d7c",
    "expires_in": 3600,
    "token_type": "bearer",
    "scope": "ifc:read
    ifc:write
    cloud:read
    cloud:manage
    document:read
    document:write"
}
```

See also:

See also *the current Authentication process*

3.4 Errors

BIMData uses conventional HTTP response codes to indicate the success or failure of an API request.

- Codes in the 2xx range indicate success.
- Codes in the 4xx range indicate an error from the client-side. Client-side has failed given the information provided (e.g., a required parameter was omitted, you don't have permission to access this resource, etc.).
- Codes in the 5xx range indicate an error with Bimdata's servers (these should be rare).

Some 4xx errors include an error code explaining the reported error:

3.4.1 400 Bad Request Response example

```
{
    "name": [
        "This field is required."
    ]
}
```

See also:

See also *the API documentation*

3.5 Filters

Many API end-points allow filtering. See available filters in the Swagger doc[7].

3.5.1 Examples

Let use the resource IFC: `/cloud/{cloud_pk}/project/{project_pk}/ifc`

The response list will only include completed IFCs (see *IFC*). You can combine several filters. Elements matching all combined filters will be returned.

Important: Filtering is an AND operation.

Note: No OR operation is supported in this version.

cURL

```
curl -X GET \
'https://api-staging.bimdata.io/cloud/1/project/1/ifc?status=C
↪' \
-H 'Authorization: Bearer ZeZr9oYxHspA8OdSCo9uftaLaEHX1N' \
-H 'Content-Type: application/json' \
```

[7] https://api-beta.bimdata.io/doc

Python

```python
import requests

url = "https://api-staging.bimdata.io/cloud/1/project/1/ifc"

querystring = {"status":"C"}

headers = {
    'Content-Type': "application/json",
    'Authorization': "Bearer ZeZr9oYxHspA8OdSCo9uftaLaEHX1N",
    }

response = requests.request("GET", url, headers=headers, ␣
↪params=querystring)

print(response.text)
```

JavaScript

```javascript
var request = require("request");

var options = { method: 'GET',
url: 'https://api-staging.bimdata.io/cloud/1/project/1/ifc',
qs: { status: 'C' },
headers:
{ Authorization: 'Bearer ZeZr9oYxHspA8OdSCo9uftaLaEHX1N',
    'Content-Type': 'application/json',
}
};

request(options, function (error, response, body) {
if (error) throw new Error(error);

console.log(body);
});
```

See also:

See also *the API documentation*

3.6 Webhooks

Webhooks let you build automation around BIMData API. Your app can subscribe to certain events on BIMData API and when one event is triggered, we'll send an HTTP POST payload to the configured URL.

Webhooks can be configured on a cloud. All projects of this cloud emits events.

3.7 Events

Each event corresponds to a set of actions.

Event	Triggered when…
bcf.topic.creation	a BCF Topic is created
bcf.topic.update	a BCF Topic is updated
bcf.comment.creation	a BCF comment is created
bcf.comment.update	a BCF comment is updated
bcf.topic.full.creation	a BCF Topic is created, send a FullTopic object
bcf.topic.full.update	a BCF Topic is updated, send a FullTopic object
ifc.process_update	the status of an IFC is changed (when it's processed)
project.update	a project is updated
project.creation	a project is created

3.8 Payload

Every payload send by BIMData API looks like `{"event_name": event_name, "cloud_id": cloud_id, "data": payload}`

Where:

- `event_name` is the name of the triggered event
- `cloud_id` is the cloud that triggered the event
- `payload` is the content of the event.

It mostly uses the same serialization than the API Models.

3.9 Ping a webhook

You can try if a webhook is well configured with https://api-next.bimdata.io/doc#/application/pingWebHook

3.10 Signature

To verify if the Webhook is sent from BIMData API and not from a malicious user, we sign out HTTP POST requests. The signature is an HMAC hex digest generated using the `sha256` hash function and the secret as the HMAC key signing the body of the request.

This signature is sent over the `x-bimdata-signature` HTTP Header.

Here is a python example to check the signature:

```
import hmac
import hashlib

def is_signed(request):
    req_signature = request.META.get("HTTP_X_BIMDATA_SIGNATURE")
    if not req_signature:
        return False

    body_signature = hmac.new(
        WEBHOOK_SECRET.encode(), request.body, hashlib.sha256
    ).hexdigest()

    return hmac.compare_digest(req_signature, body_signature)
```

3.11 Authorizations

API routes to manage Webhooks require the `webhook:manage` scope. As these calls don't involve a user, the app needs to be authorized itself on the Cloud and can't behave as a User.

See also:

See also *about IFC*

3.12 Security

We treat your data with care and apply state-of-the-art security, follwing the OWASP recommandations. Here are answers on how we keep your data secure.

3.12.1 Where is my BIMData hosted?

Your BIMData data and BIMData.io infrastructure are hosted in France by OVH. Files (DMS and IFCs) are stored with OVH Object Storage[8]

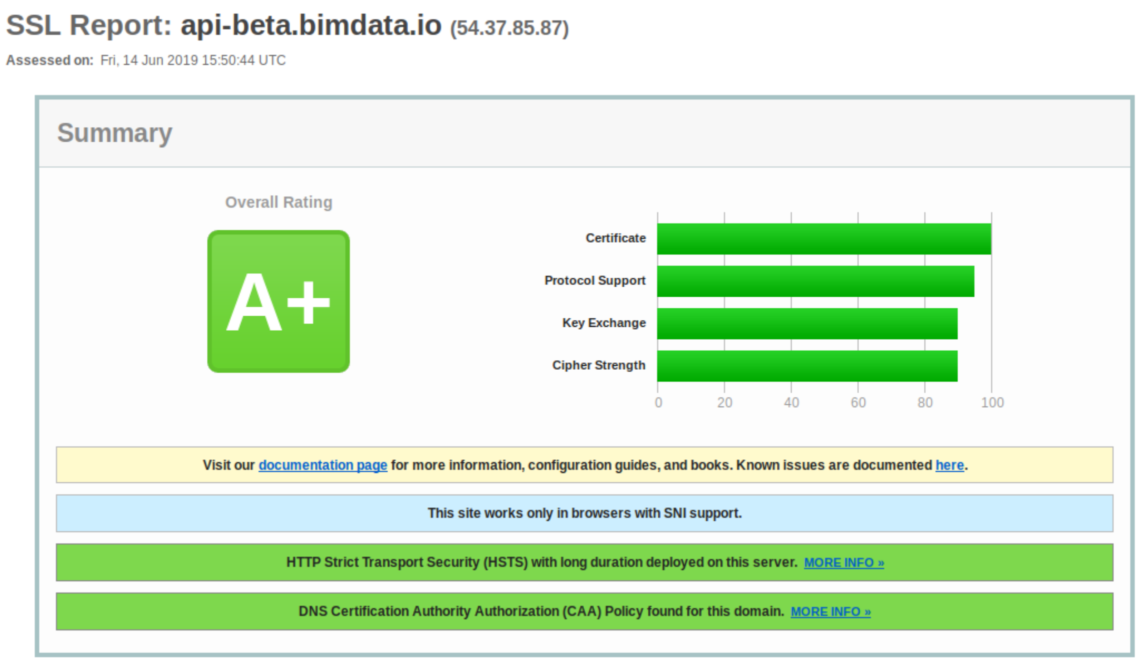
3.12.2 How often are backups made?

Your data are saved on a daily basis.

3.12.3 HTTPS

BIMData's HTTPS implementation is graded A+ on SSL Labs website.

[8] https://www.ovh.com/fr/public-cloud/object-storage/


SSL Report: api-beta.bimdata.io (54.37.85.87)

Assessed on: Fri, 14 Jun 2019 15:50:44 UTC

Summary

Overall Rating

A+

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This site works only in browsers with SNI support.

HTTP Strict Transport Security (HSTS) with long duration deployed on this server. MORE INFO »

DNS Certification Authority Authorization (CAA) Policy found for this domain. MORE INFO »

3.12.4 Where do I go if I have more questions?

Please contact us by email: <support@bimdata.io>

4 Tutorials

Getting Started

Follow the guide and make your first steps into the BIMData's API.

Viewer

Getting Started with the Viewer

API

How-to retrieve elements of your model through the API

BCF

You want statistics about which Elements are the most commented with BCF.

Guided Tour

Take a tour around and create your first application

4.1 Guided Tour

4.1.1 1 - Which app will you create?

The implementation of your app depends on your needs. You can create several types of applications.

4.1.1.1 Backend-less application

Chose this way if you are developing a backend-less application.

These applications:

- could share data with other applications like BIMData Platform or any other third-party app.
- must use **BIMData Connect users** credentials system.
- are usually mobile apps or small Javascript apps.

Create a backend-less app

4.1.1.2 Application with a backend

With BIMData Connect Users

Chose this way if your app has a backend (PHP, NodeJS, Python, .NET, etc.).

These applications:

- could share data with other applications like BIMData Platform or any other third-party app.
- must use **BIMData Connect users** credentials system.

Create a backend app

Without Users

Chose this way if you don't want to use BIMData Connect users.

These applications:

- have to manage their own users and authorizations
- can't share data with other BIMData applications
- have an easier setup

Create a backend app

4.1.2 2 - Authentication

For the authentication part, your application has to be registered on the BIMData Connect account manager (even if you're not using BIMData Connect users).

Depending on the type of app you have, you may use different ways to authenticate your app or your users.

Your goal is to retrieve an **Access Token**.

An **Access Token** is needed for every call to the BIMData's API. It represents your app, or a user using your app.

4.1.2.1 Retrieve an app Access Token

An application Access Token represents the app itself and is not linked to any user. It is used when you don't use **BIMData Connect** users or if you want to use webhooks or if you're doing some automated tasks.

Note: See Get Access Token documentation for more information.

4.1.2.2 Retrieve a user Access Token

A user Access Token represents a user using your app. It means when you're calling the API with this token, you will get the data (clouds, projects, models, etc.) the user has access to.

Before you can retrieve a user Access Token, the user must explicitly allow your application to behave as the user. There are multiple ways to ask them their consent, you can see them LINK TO OpenID Connect Authorization flows

Note: See Authentication Flow documentation

4.1.3 3 - API Onboarding

BIMData API is a tool to interact with your models stored on BIMData's servers. Using the API, you can manage your projects, the clouds, upload your IFC files, manage them and retrieve and update data from your models through endpoints.

BIMData API follows these general principles:

- All API access is over HTTPS
- All non-binary data is sent and received as JSON
- Errors are sent using standard HTTP response codes (400, 404, 403)
- Actions are indicated by HTTP verbs: GET, POST, PUT, PATCH, DELETE
- All authentication is possible via OAuth2 bearer tokens

4.1.3.1 Cloud

A cloud is a global space where your projects are hosted.

Note: To learn more about the cloud, see the concept page.

4.1.3.2 Filters

Many API end-points allow filtering.

Note: To learn more about the filters, see the concept page.

4.1.4 4 - Include the Viewer

See the dedicated page Getting Started with the Viewer

4.1.5 5 - Users Management

There are currently 3 roles defined for Users. Each User has a Role, and each User belongs to a Project.

4.1.5.1 Roles

Users can have these Roles:

- admin
- user
- guest

Constant values in API

Using the API, there are constant values associated with roles.

See the User endpoint to learn about the usage.

Note: These constants are only used in API.

When checking User's role through the API, the values are:

- **Cloud role's values**
    - admin: 100
    - user: 50
- **Project role's values**
    - admin: 100
    - user: 50
    - guest: 25

4.1.5.2 User in the Cloud

Every User in the Cloud is linked to a Project.

Admin

A cloud Admin can see every other member of the Cloud, can invite other Users as admin in the Cloud.

By default, the cloud Admin has admin rights on every project on the Cloud.

A cloud admin can ban any User from the Cloud.

Warning: Ban a User exclude the User from all Projects of the Cloud.

Member

A Cloud member is at least a member of one Project.

4.1.5.3 User in the Project

Any User in any Project can read the user list and see the other users of the project.

Admin

A Project admin can invite Users to the Project.

Note: The User is implicitly invited in the Cloud.

The Project admin manages the Roles of the Users: the admin car add, edit or delete Roles.

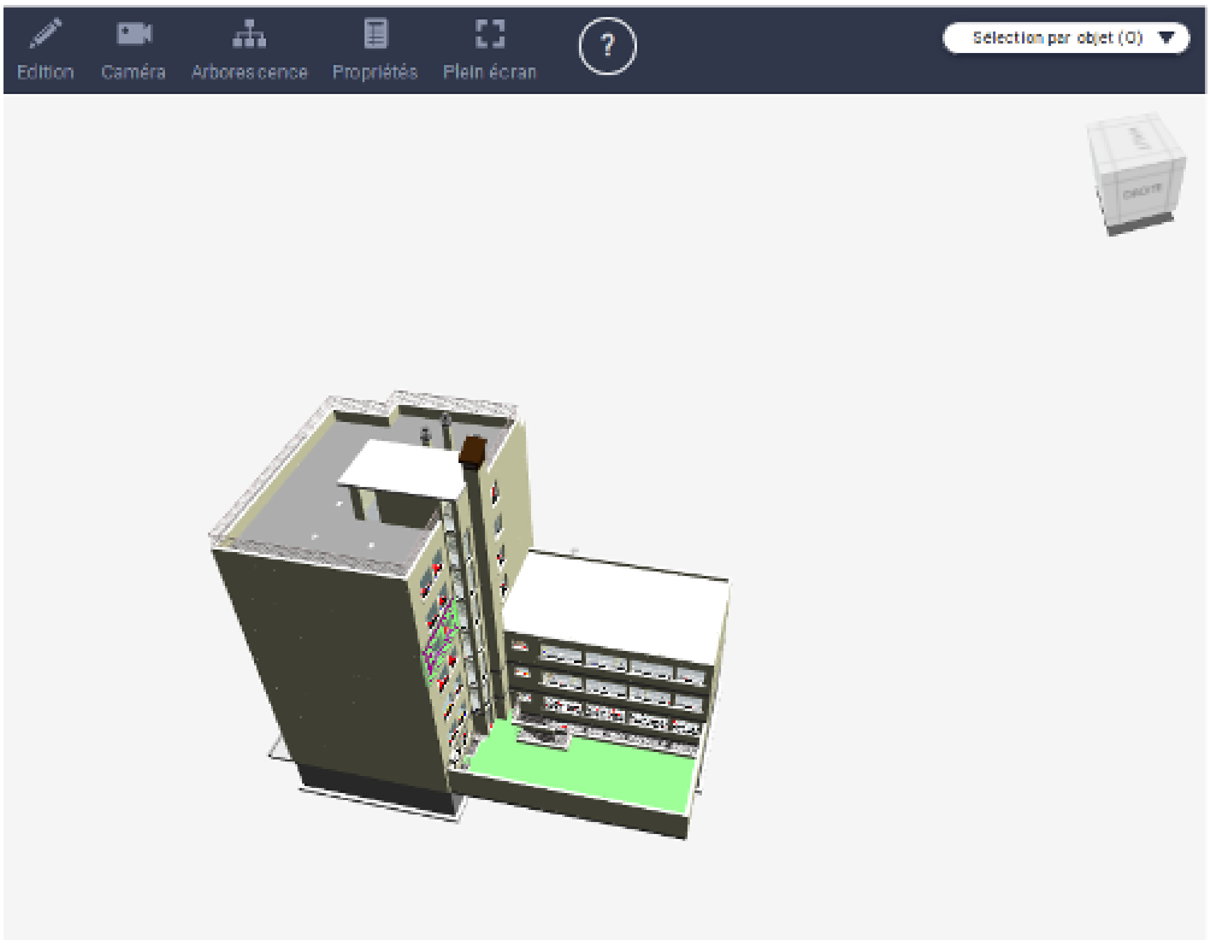
Member

Can read and write DMS, model, and BCF.

Guest

Can read-only: DMS, models, BCF and write BCF content.

4.2 Getting started with the Viewer



4.2.1 Include the Viewer

With this code in your HTML page, you load a Viewer with our Demo Model.

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
↪xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
</head>
<body>
<div id="embed" style="width: 100%; height: 100vh;"></div>
<script src="https://cdn-staging.bimdata.io/js/bimdata-viewer-embed.js">
↪</script>
<!--the viewer itself -->
<script type="text/javascript"
   src="https://unpkg.com/@bimdata/bimdata-api-client/dist/javascript-
↪api-client.min.js"></script> <!-- API call -->
<script>
```

(continues on next page)

(continued from previous page)

```
    var accessToken = 'DEMO_TOKEN';
    var cloudId = 88;
    var projectId = 100;
    var ifcId = 175;

    var viewer = new BIMDataViewer('embed', {
        accessToken: accessToken,
        cloudId: cloudId,
        projectId: projectId,
        ifcId: ifcId
    });
</script>
</body>
</html>
```

4.2.2 Minimal Viewer inclusion

With this code in your HTML page, you load an empty Viewer.

```
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
↪TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
</head>
<body>
<div id="embed" style="width: 100%; height: 100vh;"></div>
<script src="https://cdn-staging.bimdata.io/js/bimdata-viewer-embed.js">
↪ </script><!--the viewer itself-- >
<div id="embed" style="width: 100%; height: 100vh;" ></div>
<script>
    var viewer = new BIMDataViewer('embed');
</script>
</body>
</html>
```

4.2.3 Load your model

Replace the Javascript by the following code, and you load a Viewer with your Model.

```
var myAccessToken = 'DEMO_TOKEN';
var myCloudId = 88;
var myProjectId = 100;
var myIfcId = 175;
```

(continues on next page)

(continued from previous page)

```
var viewer = new BIMDataViewer('embed', {
accessToken: myAccessToken,
cloudId: myCloudId,
projectId: myProjectId,
ifcId: myIfcId
});
```

4.2.4 Customize the Viewer

See *the available methods* to customize the Viewer.

See also:

See also the Viewer's tutorials *Using the viewFit focus* and *Doors filtering example*

4.3 Retrieve elements of your model

This tutorial is about

How-to retrieve elements of your model through the API

The BIMData Platform allows you to get any information on your model. In this tutorial, you learn how to retrieve elements of your model. In this tutorial, we use the *client_credential* type of authentication, that is available when you have an application. Follow the steps and learn how to query your model through our API.

4.3.1 Step 1. Get your token

To get an Access Token, you first create an application named "Windows retriever". Then follow the procedure described on the dedicated page to get your Access Token. You are now ready to use the API.

4.3.2 Step 2. Set up your project

Once your app exists and you have your Access Token, you are able to use the API. Begin with the creation of a Cloud (What's a cloud?), in which you create a Project (What's a project?). In the script below, there is an example of the creation of a project in your Cloud through API, so you can have a `projectId`. First, define a name to create your first Cloud. Post this `name` on https://api-staging.bimdata.io/cloud using your Access Token. Then use the `cloudId` to create your first Project.

```
import requests

#previous step gave you an access_token
#prepare headers and cloud_name for POST request
headers = {
    'authorization': 'Bearer '+ access_token,
}
cloud_name = {
    'name': 'Windows retrieval'
}
response = requests.post(f'https://api-staging.bimdata.io/cloud', ␣
↪data=cloud_name, headers=headers)
assert response.status_code == 201

cloud_id = response.json().get('id')
response = requests.post(f'https://api-staging.bimdata.io/cloud/{cloud_
↪id}/project', headers=headers)
assert response.status_code == 201

project_id = response.json().get('id')
```

Let's upload your IFC model!

4.3.3 Step 3. Upload your IFC

The API let you upload your IFC file. In this tutorial, you can use this IFC file: Download Cassiopea IFC[9]

4.3.3.1 Use the API to upload

Use the `/cloud/{cloud_pk}/project/{project_pk}/document` route to upload your file. The documentation for this route is available on our API Reference page.

```
import requests

#prepare headers and payload for POST request
headers = {
    'authorization': 'Bearer '+ access_token,
}
payload = {
    'name': 'My lovely IFC'
}
my_ifc_to_upload = {'file': open('/path/to/XXX.ifc', 'rb')}

#previous script gives you the cloud_id and the project_id
```

(continues on next page)

[9] https://drive.google.com/file/d/1njhweVCFvDNl8Gy3B1HxAolcfExt0Tg-/view?usp=sharing

(continued from previous page)

```
url = f'https://api-staging.bimdata.io/cloud/{cloud_id}/project/{project_
↪id}/document'
response = requests.post(url, data=payload, files=my_ifc_to_upload,␣
↪headers=headers)
assert response.status_code == 201

my_ifc_id = response.json().get('ifc_id')
```

4.3.3.2 Follow the upload process

During the upload, you must query the server to get information about the upload process, see the IFC documentation page about the available information. The server detects IFC format and you can get information about your file using the API: https://api-staging.bimdata.io/doc#/ifc/getIfc

Note: The IFC document provided in this tutorial takes approximatively 10 seconds to be processed.nUsually, the processing time could be very different depending on the IFC file.

```
import time
import requests

ready = False

while not ready:
    url = f'https://api-staging.bimdata.io/cloud/{cloud_id}/project/
↪{project_id}/ifc/{my_ifc_id}'
    response = requests.get(url, headers=headers)
    assert response.status_code == 200

    status = response.json().get('status')

    if('C' == status):
        ready = True
        #your IFC is ready to query
    else:
        #print('not ready yet')
        time.sleep(1)
```

When the status is *C* meaning Complete, your IFC document is uploaded and processed. Let's use the BIMData API to query your model!

4.3.4 Step 4. Retrieve windows

In this tutorial, you want *all the windows of the building* described in your IFC.

4.3.4.1 Retrieve elements

The route is: */cloud/{cloud_pk}/project/{project_pk}/ifc/{ifc_pk}/element*

As listed on the documentation page for this route: the mandatory parameters are:

- *cloud_pk* string
- *ifc_pk* string
- *project_pk* string

4.3.4.2 Use filters

In addition, you can filter by:

- *type* string
- *classification* string
- *classification__notation* string

To retrieve only windows, the accurate filter is *type*: **IfcWindow**. You get a list of windows, all the windows of your model.

```
import requests
# This script requires an IFC document uploaded

my_filter = {
    'type': 'IfcWindow'
}
url = f'https://api-staging.bimdata.io/cloud/{cloud_id}/project/{project_
↪id}/document/{my_ifc_id}'
response = requests.get(url, data=my_filter, headers=headers)
assert response.status_code == 200

all_windows = response.json()
#all_windows are available in this var for your next scripts
```

With the filters, every IFC element can be retrieved. You can retrieve any element in the collection provided in the API.

See also:

See also *the API completedocumentation*

4.4 Process BCF data for Excel export

You want statistics about which Elements are the most commented with BCF. You have to export data about the commented elements, you want to produce Excel files to import to a stats software. You will create a CSV file, each line is for an Element. Our API implements the 2.1 version of the BCF specification[10]

[10] https://github.com/buildingSMART/BCF-API/tree/v2.1

The column are:

- Element Type
- Classification title
- Classification Notation
- Element UUID
- Topic GUID

Let's help you to achieve your goal.

Note: Performance

For clarity reasons, the code shown below is not optimized.

4.4.1 Step 1 - Get all Topics from your Model

First step, with a single endpoint, you get all the Topics attached to your Model, and the IFCs ID. Iterate on topics and do another request to get the Viewpoints.

Tip: API specs Explore the OpenAPI specification for the endpoint: https://api-staging.bimdata.io/doc?format=openapi

Note: The BIMData's UI tools let you have only one Viewpoint by Topic. The API let you manage several. It's the reason you see `viewpoints[0]` line 13.

```
1  url = f'https://api-staging.bimdata.io/bcf/2.1/projects/{project_pk}/topics'
2  response = requests.get(url, headers=headers)
3  assert response.status_code == 200
4  for topic in response.json():
5      #iterate on topics and do another request to get viewpoints
6      url = f'https://api-staging.bimdata.io/bcf/2.1/projects/{project_pk}/topics/{topic["guid"]}/topic-viewpoints'
7      response = requests.get(url, headers=headers)
8      viewpoints = response.json()
9  
10     if not viewpoints: #no viewpoint found
11         continue
12 
13     viewpoint = viewpoints[0]
14     topic_guid = topic.get("guid")
15     ifcs_list = topic.get("ifcs")
16 
```

(continues on next page)

(continued from previous page)

```
    #populate elements list
    elements = viewpoint["components"]["selection"]
    element_uuids = [element["ifc_guid"] for element in elements]
```

4.4.2 Step 2 - Get the Elements

Then you iterate on the Viewpoint's elements, and in this loop on the IFCs to get the details for each element, such as classification.

Warning: Some IFCs could not have the Element, in the case of multi-model BCF.

```
allElements = []
for element_uuid in element_uuids:
    for ifc_pk in ifcs_list:
        url = f'https://api-staging.bimdata.io/cloud/{cloud_pk}/project/
↪{project_pk}/ifc/{ifc_pk}/element/{element_uuid}'
        response = requests.get(url, headers=headers)

        #if there is no matching IFC, we continue with the next one
        if(response.status_code != 200):
            continue

        raw_element = response.json()
        for classification in raw_element["classifications"]:
            element = {
                "type": raw_element["type"],
                "classification_title": classification["title"],
                "classification_notation": classification["notation"],
                "uuid": element_uuid,
                "topic_guid": topic_guid
            }
            allElements.append(element)
```

4.4.3 Step 3 - Write your file

For our example, this is the code you write:

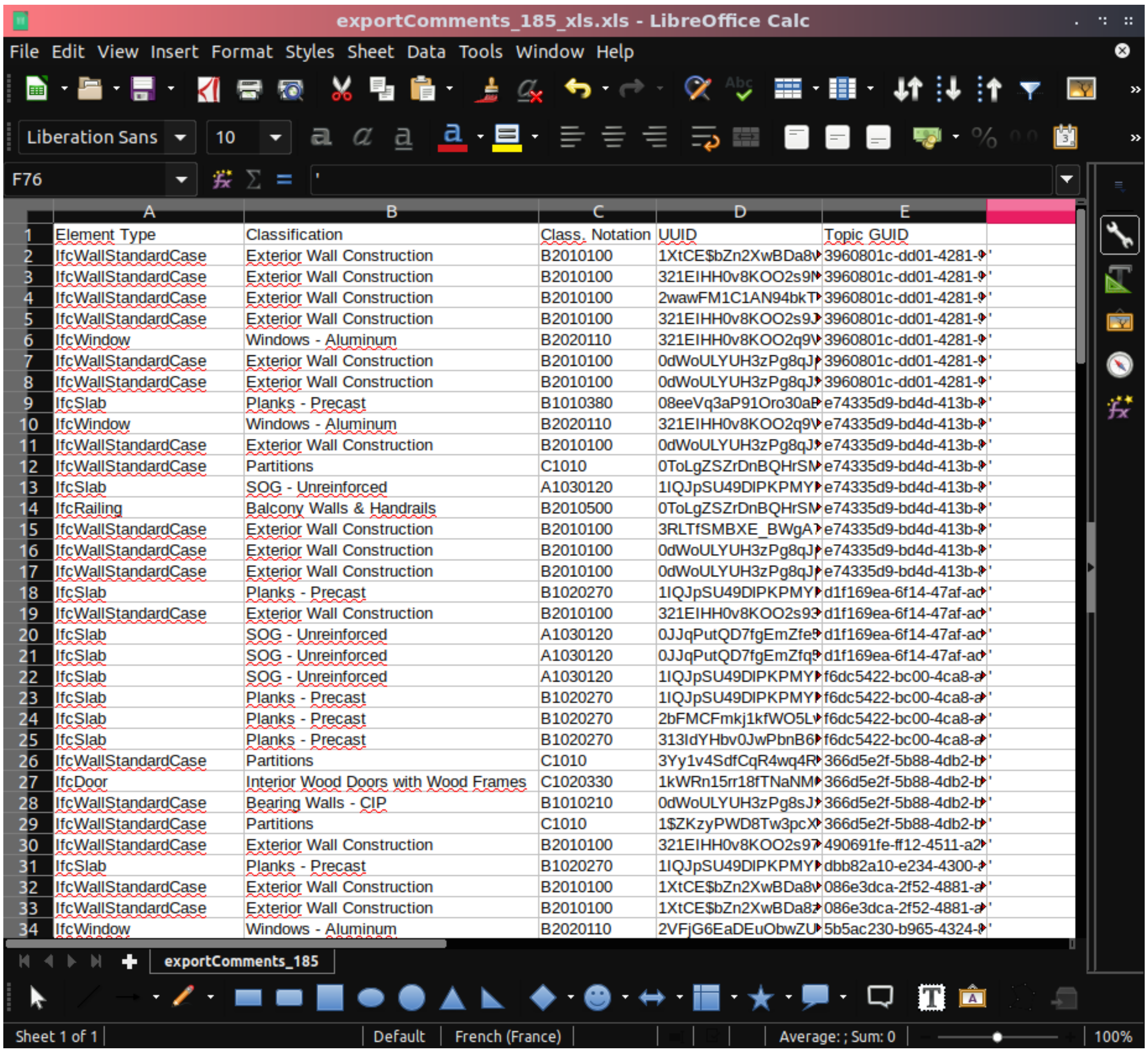
```
with open(f'exportComments_{project_pk}.csv', 'w', newline='') as␣
↪csvfile:
        topicwriter = csv.writer(csvfile, delimiter=';',
                                quotechar='"', quoting=csv.QUOTE_
↪MINIMAL)
        topicwriter.writerow(["Element Type", "Classification", "Class.␣
↪Notation", "UUID", "Topic GUID"]) #changeHeaders
```

(continues on next page)

(continued from previous page)

```
        for oneElement in allElements:
            topicwriter.writerow([
                oneElement["type"],
                oneElement["classification_title"],
                oneElement["classification_notation"],
                oneElement["uuid"],
                oneElement["topic_guid"]
            ])
```

You have now a CSV file Excel-compatible!


exportComments_185_xls.xls - LibreOffice Calc

F76 '

	A	B	C	D	E	
1	Element Type	Classification	Class. Notation	UUID	Topic GUID	
2	IfcWallStandardCase	Exterior Wall Construction	B2010100	1XtCE$bZn2XwBDa8	3960801c-dd01-4281-	'
3	IfcWallStandardCase	Exterior Wall Construction	B2010100	321EIHH0v8KOO2s9	3960801c-dd01-4281-	'
4	IfcWallStandardCase	Exterior Wall Construction	B2010100	2wawFM1C1AN94bkT	3960801c-dd01-4281-	'
5	IfcWallStandardCase	Exterior Wall Construction	B2010100	321EIHH0v8KOO2s9	3960801c-dd01-4281-	'
6	IfcWindow	Windows - Aluminum	B2020110	321EIHH0v8KOO2q9	3960801c-dd01-4281-	'
7	IfcWallStandardCase	Exterior Wall Construction	B2010100	0dWoULYUH3zPg8qJ	3960801c-dd01-4281-	'
8	IfcWallStandardCase	Exterior Wall Construction	B2010100	0dWoULYUH3zPg8qJ	3960801c-dd01-4281-	'
9	IfcSlab	Planks - Precast	B1010380	08eeVq3aP91Oro30a	e74335d9-bd4d-413b-	'
10	IfcWindow	Windows - Aluminum	B2020110	321EIHH0v8KOO2q9	e74335d9-bd4d-413b-	'
11	IfcWallStandardCase	Exterior Wall Construction	B2010100	0dWoULYUH3zPg8qJ	e74335d9-bd4d-413b-	'
12	IfcWallStandardCase	Partitions	C1010	0ToLgZSZrDnBQHrS	e74335d9-bd4d-413b-	'
13	IfcSlab	SOG - Unreinforced	A1030120	1IQJpSU49DIPKPMY	e74335d9-bd4d-413b-	'
14	IfcRailing	Balcony Walls & Handrails	B2010500	0ToLgZSZrDnBQHrS	e74335d9-bd4d-413b-	'
15	IfcWallStandardCase	Exterior Wall Construction	B2010100	3RLTfSMBXE_BWgA	e74335d9-bd4d-413b-	'
16	IfcWallStandardCase	Exterior Wall Construction	B2010100	0dWoULYUH3zPg8qJ	e74335d9-bd4d-413b-	'
17	IfcWallStandardCase	Exterior Wall Construction	B2010100	0dWoULYUH3zPg8qJ	e74335d9-bd4d-413b-	'
18	IfcSlab	Planks - Precast	B1020270	1IQJpSU49DIPKPMY	d1f169ea-6f14-47af-a	'
19	IfcWallStandardCase	Exterior Wall Construction	B2010100	321EIHH0v8KOO2s9	d1f169ea-6f14-47af-a	'
20	IfcSlab	SOG - Unreinforced	A1030120	0JJqPutQD7fgEmZfe	d1f169ea-6f14-47af-a	'
21	IfcSlab	SOG - Unreinforced	A1030120	0JJqPutQD7fgEmZfq	d1f169ea-6f14-47af-a	'
22	IfcSlab	SOG - Unreinforced	A1030120	1IQJpSU49DIPKPMY	f6dc5422-bc00-4ca8-	'
23	IfcSlab	Planks - Precast	B1020270	1IQJpSU49DIPKPMY	f6dc5422-bc00-4ca8-	'
24	IfcSlab	Planks - Precast	B1020270	2bFMCFmkj1kfWO5L	f6dc5422-bc00-4ca8-	'
25	IfcSlab	Planks - Precast	B1020270	313IdYHbv0JwPbnB6	f6dc5422-bc00-4ca8-	'
26	IfcWallStandardCase	Partitions	C1010	3Yy1v4SdfCqR4wq4R	366d5e2f-5b88-4db2-	'
27	IfcDoor	Interior Wood Doors with Wood Frames	C1020330	1kWRn15rr18fTNaNM	366d5e2f-5b88-4db2-	'
28	IfcWallStandardCase	Bearing Walls - CIP	B1010210	0dWoULYUH3zPg8sJ	366d5e2f-5b88-4db2-	'
29	IfcWallStandardCase	Partitions	C1010	1$ZKzyPWD8Tw3pcX	366d5e2f-5b88-4db2-	'
30	IfcWallStandardCase	Exterior Wall Construction	B2010100	321EIHH0v8KOO2s9	490691fe-ff12-4511-a2	'
31	IfcSlab	Planks - Precast	B1020270	1IQJpSU49DIPKPMY	dbb82a10-e234-4300-	'
32	IfcWallStandardCase	Exterior Wall Construction	B2010100	1XtCE$bZn2XwBDa8	086e3dca-2f52-4881-	'
33	IfcWallStandardCase	Exterior Wall Construction	B2010100	1XtCE$bZn2XwBDa8	086e3dca-2f52-4881-	'
34	IfcWindow	Windows - Aluminum	B2020110	2VFjG6EaDEuObwZU	5b5ac230-b965-4324-	'

exportComments_185

Sheet 1 of 1 | Default | French (France) | Average: ; Sum: 0 | 100%

File Edit View Insert Format Styles Sheet Data Tools Window Help
Liberation Sans
10

4.4.4 The full script

```
import requests
import csv

```

(continues on next page)

(continued from previous page)

```
cloud_pk = CLOUD_ID
project_pk = PROJECT_ID
headers = {
    "Authorization": "Bearer ACCESS_TOKEN"
}

allElements = []

url = f'https://api-staging.bimdata.io/bcf/2.1/projects/{project_pk}/
↪topics'
response = requests.get(url, headers=headers)
assert response.status_code == 200

for topic in response.json():
    url = f'https://api-staging.bimdata.io/bcf/2.1/projects/{project_pk}
↪/topics/{topic["guid"]}/topic-viewpoints'
    response = requests.get(url, headers=headers)
    viewpoints = response.json()
    if not viewpoints:
        continue
    viewpoint = viewpoints[0]
    topic_guid = topic.get("guid")
    ifcs_list = topic.get("ifcs")

    elements = viewpoint["components"]["selection"]
    element_uuids = [element["ifc_guid"] for element in elements]

    for element_uuid in element_uuids:
        for ifc_pk in ifcs_list:
            url = f'https://api-staging.bimdata.io/cloud/{cloud_pk}/
↪project/{project_pk}/ifc/{ifc_pk}/element/{element_uuid}'
            response = requests.get(url, headers=headers)
            if(response.status_code != 200):
                continue

            raw_element = response.json()
            for classification in raw_element["classifications"]:
                element = {
                    "type": raw_element["type"],
                    "classification_title": classification["title"],
                    "classification_notation": classification["notation
↪"],
                    "uuid": element_uuid,
                    "topic_guid": topic_guid
                }
                allElements.append(element)
```

(continues on next page)

(continued from previous page)

```

with open(f'exportComments_{project_pk}.csv', 'w', newline='') as csvfile:
        topicwriter = csv.writer(csvfile, delimiter=';',
                                quotechar='"', quoting=csv.QUOTE_MINIMAL)
        topicwriter.writerow(["Element Type", "Classification", "Class. Notation", "UUID", "Topic GUID"])
        for oneElement in allElements:
            topicwriter.writerow([
                oneElement["type"],
                oneElement["classification_title"],
                oneElement["classification_notation"],
                oneElement["uuid"],
                oneElement["topic_guid"]
            ])
```

You now have your data ready to be printed in a CSV file, or sent to your favorite Excel-file generator.

See also:

See also *more about IFC*

5 Cookbook

Create an application

How-To create your application on BIMData Connect

Create a backend-less application

How-To create a mobile or tablet application on BIMData Connect

Create a backend application

How-To create an application on BIMData Connect

Get Access Token

The script regarding the Access Token

5.1 Create your application

How-To create your application on BIMData Connect

Create an account on the https://login-staging.bimdata.io website. After the login step, go to “Manage your application” screen and fill the form to Create an Application.

You will choose a *Name* for your application, let’s type **“Wonderful app”** in the field *Name*.

Note: All the fields may be edited later.

Select the **Confidential** option in the field *Client Type*. Select the **code (Authorization Code Flow)** option in the field *Response Type*.

These settings allow your app to communicate using a unique Token Access. Other choices are useful to manage the access rights for every API call.

5.1.1 Response Type

See the three Types of Auth to learn more about the Response Type.

In the field *Redirect URIs*, set at least the value `http://localhost`: for now, you want your brand new application to receive any response.

5.1.2 Fields description

Table 1: Fields description

Field name	Description
Name	sets the name of your application. This name will be shown to the Users you will invite.
Client Type	**sets the confidentiality of your application's credentials** • A confidential client is an application that is capable of keeping a client password confidential to the world. • A public client is an application that is not capable of keeping a client password confidential. • See the Tutorial made by Jenkov[11] for more explanations.
Response Type	sets the way of authenticating, see the Authentication documentation content to learn more.
Scopes	lists every granted access for your application on the data. See the Scopes documentation content to learn more.
JWT Algorithm	lets you choose which algorithm is used to encode ID tokens
Redirect URIs	determines where the authorization server sends a response to your app. It's a list of authorized URIs where the user can be redirected after the authorization process
Post Logout Redirect URIs	list of callback URIs when the user logs out from your application
Website URL	Your application's website, if applicable. Visible on the BIMData Connect platform to users.
Terms URL	Terms of service of your application, if applicable. Visible on the BIMData Connect platform to users.
Logo Image	Logo of your application, if applicable. Visible on the BIMData Connect platform to users.

[11] http://tutorials.jenkov.com/oauth2/client-types.html

Let the other fields to their default value and submit the form. You created your first application.

You will see 2 new pieces of information: the Client ID and the Client Secret. This Client ID and Client Secret are mandatory to build your application.

See also:

See also *about Security*

5.2 How-to get your Access Token

Warning: Pre-requisite: having created an application

Look at the procedure to create an application, then come back here. After the application is created, you can retrieve it on the BIMData.io account manager, in the “Manage your application” screen.

5.2.1 Get your access token

The Client ID and the Client Secret are the 2 elements that you need to get the Access Token of your application from the Authentication Server. You will need this Access Token for every call of the bimdata’s API.

5.2.1.1 Python

```
import requests

payload = {
    "client_id": CLIENT_ID,
    "client_secret": CLIENT_SECRET,
    "grant_type": "client_credentials",
}

#Get the token
response = requests.post("https://login-staging.bimdata.io/token",␣
↪data=payload)
access_token = response.json().get("access_token")
```

5.2.1.2 Curl

```
curl --request POST "https://connect-next.bimdata.io/token" \
  --header "Content-Type: application/x-www-form-urlencoded" \
  --data "grant_type=client_credentials&client_id=YOUR_CLIENT_ID&client_
↪secret=YOUR_CLIENT_SECRET"
```

Warning: This API call doesn't accept JSON

Be sure to use application/x-www-form-urlencoded encoding

See also:

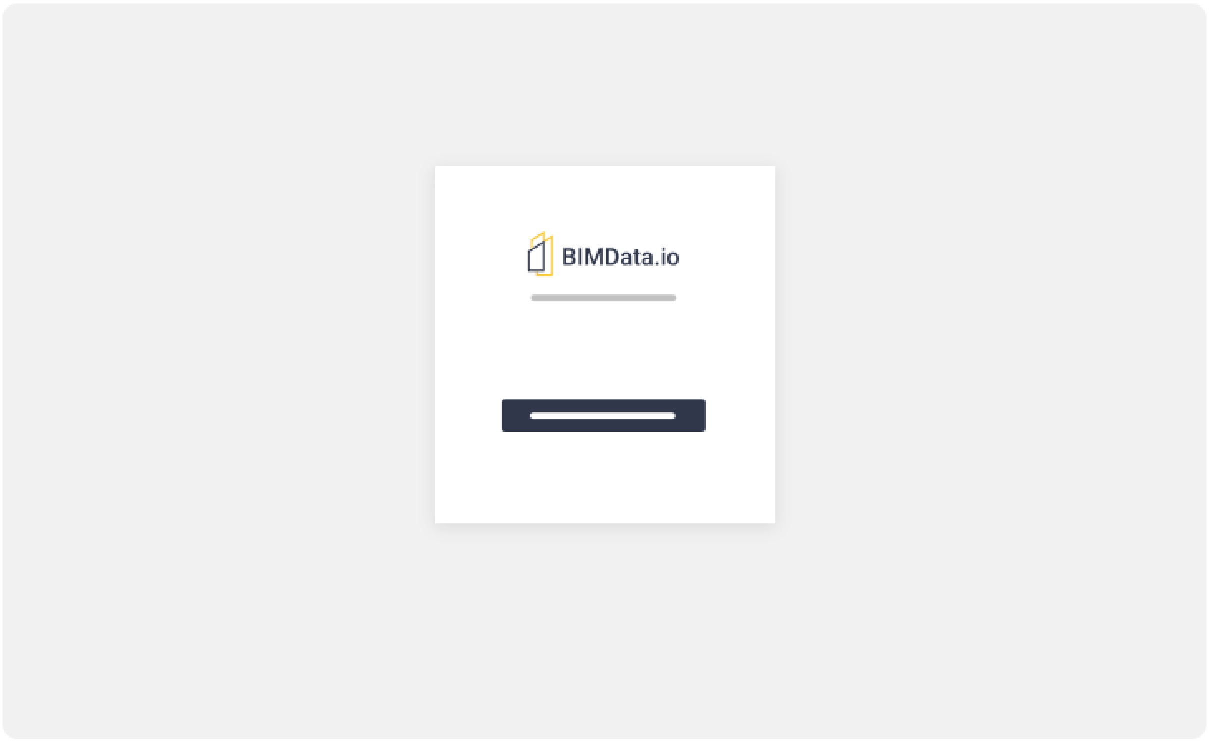
See also *the API documentation*

6 Platform User Guide

The Platform let you have as many Clouds as you need. In every Cloud, you can create as many Projects as you want.

Every Project is a place to share about a BIM project: in a Project, you could be Admin, User or Guest. The features you can use could be limited by your Role on the Project.

6.1 Getting Started



Log in the Platform on https://platform-staging.bimdata.io

Once logged in, you can navigate through Clouds and Projects directly with the Choice List on the top of every page.

And *create your first Project*, *upload a Model*. *Invite your colleagues* and begin to collaborate around your BIM Project.

6.2 Learn how-to...

Add Content

How-To add content: models, clouds, projects

Collaborate

How-To collaborate with your colleagues

Organize

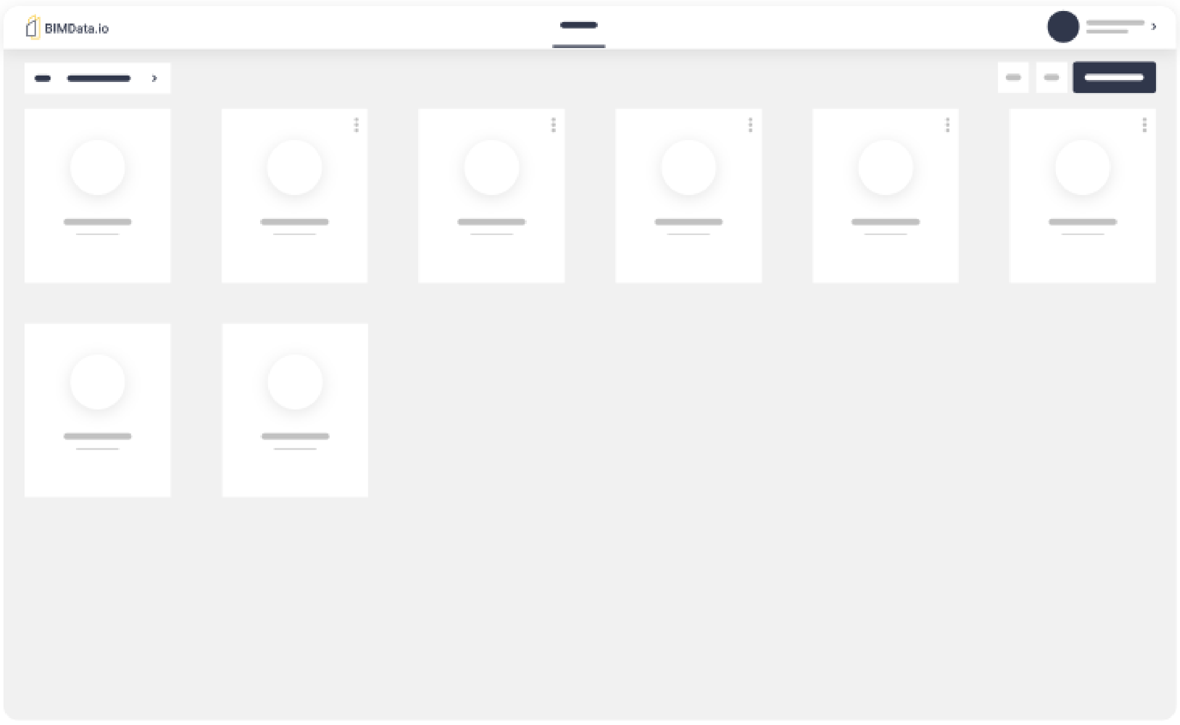
How-To organize your content: models, clouds, projects

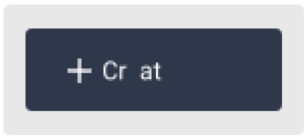
Support

6.2.1 Add Content

Learn how-to

6.2.1.1 Create a Cloud



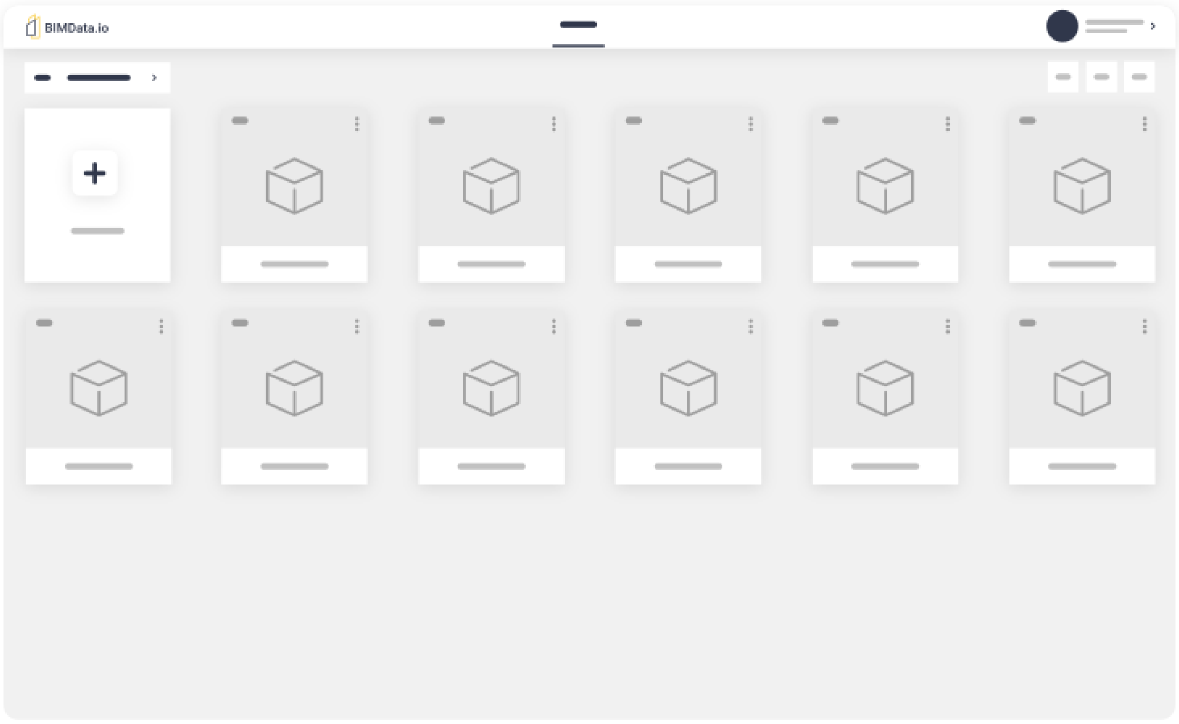
- Click [Create Cloud] 

 from the button zone
- Fill the Cloud's name

The freshly created Cloud is added at the end of the list. On each Cloud's card, the number of Projects and the number of Users are visible.

For any new Cloud, you can set Users rights. See Users Management screen for more information.

6.2.1.2 Create a Project



- Click on [New Project] on the left of your Project list
- Fill the name of your next Project

You are ready to add content in the Project's screen.

6.2.1.3 Upload your content

Add your Models

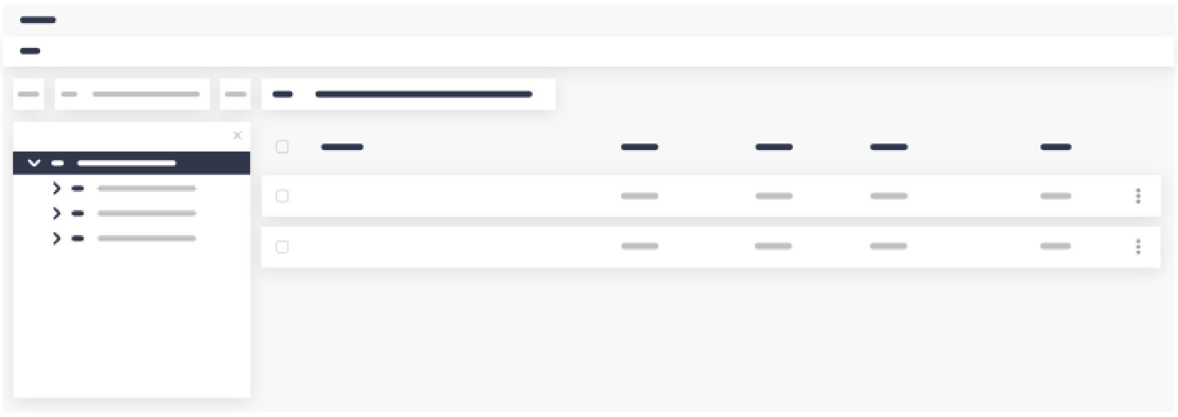
-

Click [File], above the User's List
- Chose one or several file(s)
- Click [Upload 1 file]

Your file is listed in the IFC list, with the Creator and the uploaded Status.

Add your Documents

- Click [Import a Document]
- Drop or chose a file, or several at once, from your computer
- Click [Upload 1 file]



The uploaded file is available in the Document List. You can organize it by creating Folders.

Note: For more information about the organization of file, see the Organize chapter.

6.2.1.4 Add BCF

From any Project Card, a click [Shortcut Viewer] launches the Viewer directly in full-screen mode.

From the Project Page, click [Viewer] and the Viewer opens in full-screen mode as well.

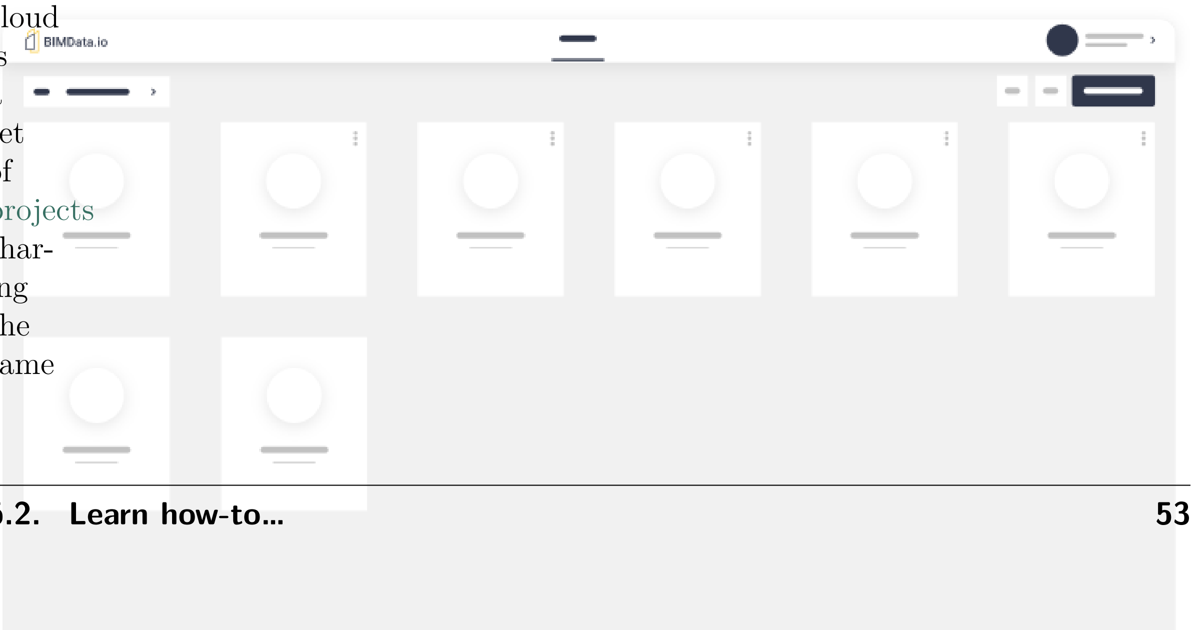
From the Viewer, every member of the Project can add BCF comments directly and collaborate.

6.2.2 Organize

Once you are logged in BIMData Connect, you can create Clouds on the Platform. In each Cloud, you can create Projects and manage diverse Documents.

6.2.2.1 Clouds

A cloud is a set of projects sharing the same



con-
fig-
u-
ra-
tion.

At
your
first
log
in,
you have already 2 Clouds existing: * Demo Cloud * your own personal Cloud

Note: The Demo Cloud is a complete example and contains a Demo Project already populated with an IFC Model.

From the main screen of the BIMData Platform, using the buttons, you can:

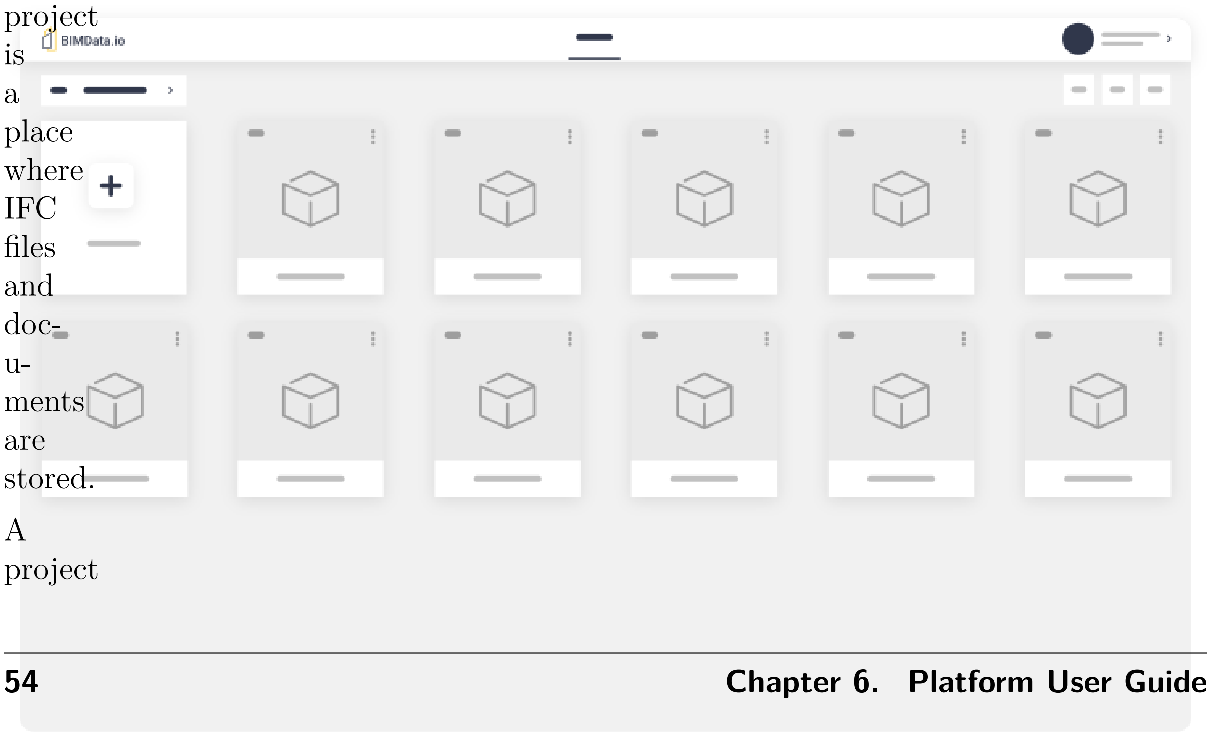
- search Clouds by name : the search is case-insensitive
- re-order Cloud by name
- create a new Cloud
- remove a Cloud

Note: The Cloud card let you know how many projects are in the Cloud, and how many Users have access to this Cloud.

6.2.2.2 Projects

A
project
is
a
place
where
IFC
files
and
doc-
u-
ments
are
stored.

A
project



be-
longs
in
a
Cloud.

From your Projects List screen, the Choice List let you navigate through your Clouds and your Projects.

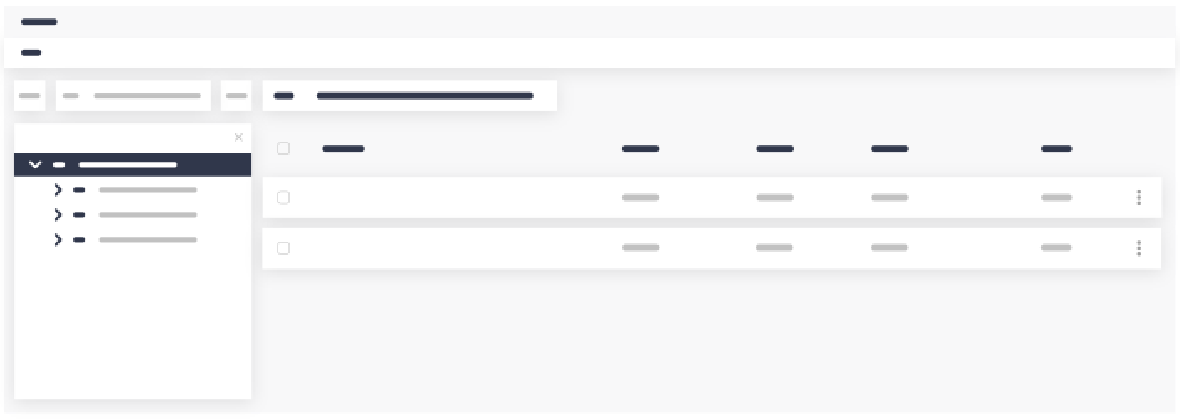
From your Projects List screen, using the buttons, you can:

- search Projects by name : the search is case-insensitive
- re-order Projects by name
- rename an existing Project
- create a new Project
- remove a Project

On the Project's screen, the Viewer shows you information about your Models: Model Preview, Map with the surrounding, User's list, Documents of the Project. From the Project's screen, you can invite Users (for more info, see Collaborate chapter), add Documents (see Add Content).

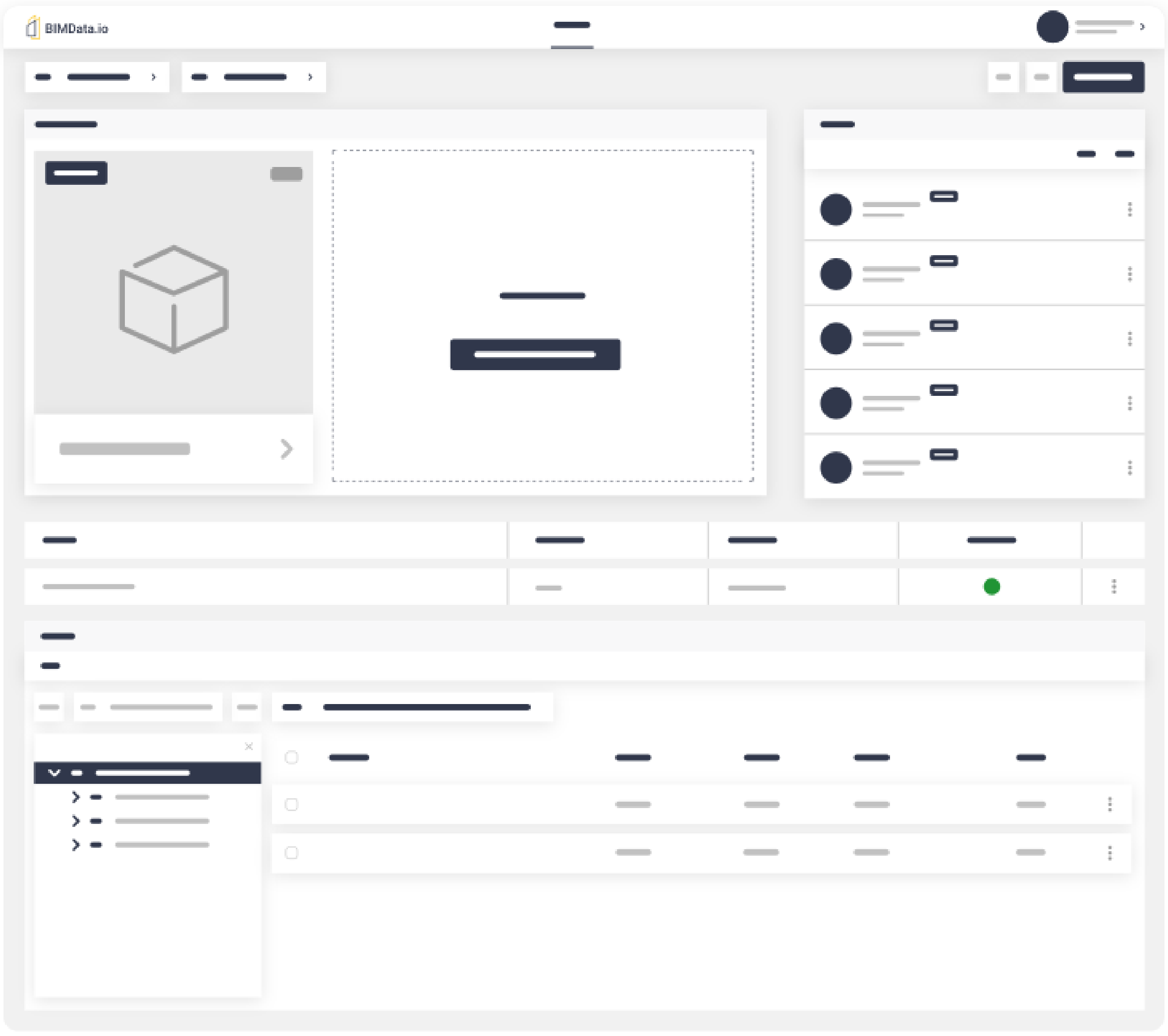
You can create directories to store and organize your Documents.

6.2.2.3 Documents

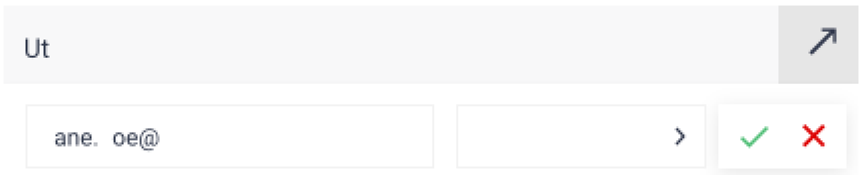


IFC files and documents can be uploaded and organized, checkplans are defined.

6.2.3 Collaborate



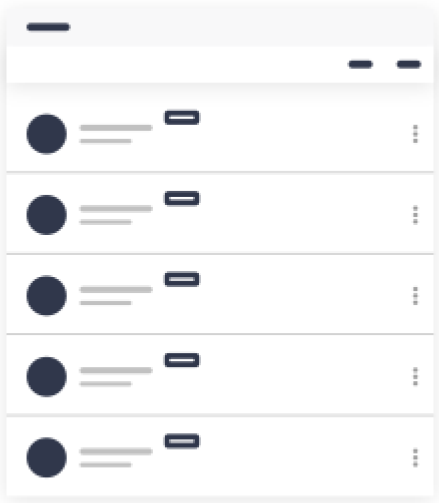
6.2.3.1 Invitations



- Click [Invite] icon on top of the Users List
- Fill in the e-mail address
- Chose the access rights: Admin, User or Guest for the Project. You could change the Role of any User later.
- Click [Validate]

The invitation is sent.

Note: Users access levels



For more information about different access rights, see the Guided Tour about Roles chapter.

Warning: Invite someone with an Admin Role invites implicitly the person in the Cloud.

Users Roles

Warning: Any User in any Project can read the User List and see the other Users of the project.

Admin

A Project admin can invite Users to the Project. A cloud Admin can see every other member of the Cloud, can invite other Users as admin in the Cloud.

The Project admin manages the Roles of the Users: the admin can add, edit or delete Roles.

Member

Can read and write Documents, Models, and BCF.

Guest

Can read-only: DMS, models, BCF and write BCF content.

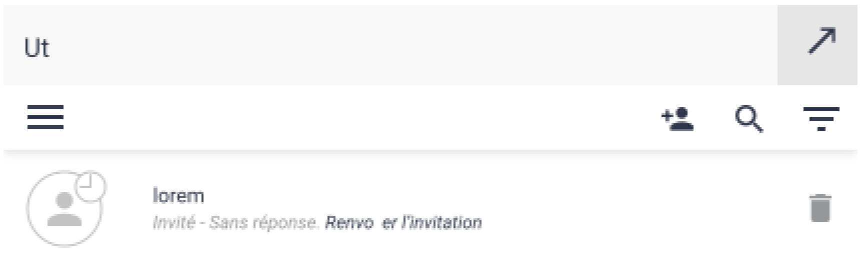
Details about invitations

Note: Invite

See the Collaborate section about Invitation, to learn about Invitation basics.

Until your recipient accepts it, the invitation appears incomplete. The e-mail contains a link to accept directly the invitation.

Tip: Invitation lost? To resend the invitation, click [Resend Invitation?]. The e-mail is sent again.


Ut

lorem
Invité - Sans réponse. Renvo er l'invitation

6.2.3.2 Revoke members

Warning: Ban a User exclude the User from all Projects of the Cloud.

To revoke a member, click [Delete User] and confirm your action. The User has no longer access to your Project.

6.2.4 Get Help

Do you need help?

6.2.4.1 Contact us

You can reach us by e-mail: support@bimdata.io

7 Viewer Documentation

BIMData provides a 3D Viewer with which you can interact with Javascript.
Getting started

7.1 Guide

- *Include the Viewer in your app*

7.2 Cookbook

Usage examples of the Viewer:

- How-to: *init the Viewer*
- How-to: *doors filtering*
- How-to: *use the viewFit focus*

7.3 Reference

- *Javascript methods of the Viewer*

7.4 Example of the Viewer

```
var accessToken = 'DEMO_TOKEN';
var cloudId = 88;
var projectId = 100;
var ifcId = 175;

var viewer = new BIMDataViewer('embed', {
  accessToken: accessToken,
  cloudId: cloudId,
  projectId: projectId,
  ifcId: ifcId
});
```

7.5 Requirements

Only WebGL-enabled browsers[12] are supported:

- Firefox
- Chrome
- Edge
- Safari 11 or later

Note: Test the WebGL support[13] of your browser.

7.5.1 Recommended

For best performance, use the latest versions of modern browsers: Firefox, Chrome, Edge or Safari. In addition to that, we recommend:

- 4 GB RAM (1GB for the system, 1GB for the browser and 2GB for the Viewer)
- Javascript and built-in hardware acceleration enabled
- Modern browser with ECMAScript 2015 support[14]

7.5.1.1 Include the Viewer in your app

To support the Viewer you will need a `<div>` element, with an id attribute, and 3 scripts:

- The JS viewer to embed with the `<script>` tag
- The API Javascript client, for API calls, to embed with the script tag
- Your own JS script: in this documentation, this part will be directly in the HTML `<script>` tag

```
<script src="https://cdn-staging.bimdata.io/js/bimdata-viewer-embed.js" >
↪<!--the viewer itself --></script>
<script>
```

The <div> element

You `<div>` is simply styled by taking the whole screen width and height. An ID will ensure the easy access in Javascript scripts.

```
<div id="embed" style="width: 100%; height: 100vh;" ></div>
```

[12] https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API
[13] https://get.webgl.org
[14] https://kangax.github.io/compat-table/es6

JS Viewer script

The Viewer is based on XeoGL Engine[15], see the *list of the methods of the Viewer* to interact with the Viewer.

The Viewer is the object BIMDataViewer. The construction of a new BIMDataViewer expects an HTMLElement or a string (ID of this HTMLElement). You will use the <div> ID.

Your own Javascript script

You can now interact with the Viewer in your script.

7.5.1.2 Example: doors filtering

BIMData provides a 3D Viewer with which you can interact with Javascript.

In this example, you want to check if your doors let the disabled person access to the building. To evaluate the number of doors that are ok or not ok, you will display all the doors of the model with a color-code according to the minimal width.

The script has 3 parts:

- you get the model from the cloud
- the Viewer load the model and you get all doors
- you hide everything, first then you set a color and show only the doors.

Get the model from the cloud

This part is covered by the Recipe "*How-to get the model from the cloud*"

Get all the doors

Using the `ifcApi` you retrieve the doors the element *IfcDoor*, the getElements() methods could take an argument to filter by IFC element. Then you enlist the doors in two separate lists, based on their width:

Retrieve the whole script on our Codepen[16].

Color doors

In this example, you first set all elements as *unpickable*, and you `ghost()` them. Then you `unghost()` and set *pickable* again only the doors elements. Since you have the doors list, you set the color of the elements related to their width.

[15] http://xeogl.org/
[16] https://codepen.io/bimdata

Example of the Viewer:

```
var accessToken = 'DEMO_TOKEN';
var cloudId = 88;
var projectId = 100;
var ifcId = 175;
var defaultClient = bimdata.ApiClient.instance;

defaultClient.basePath = 'https://api-beta.bimdata.io';
// Configure API key authorization: Bearer
var Bearer = defaultClient.authentications['Bearer'];
Bearer.apiKey = 'Bearer ' + accessToken;

var viewer = new BIMDataViewer('embed', {
  accessToken: accessToken,
  cloudId: cloudId,
  projectId: projectId,
  ifcId: ifcId
});

// Wait the viewer to finish loading
viewer.on('viewer-load', function(e) {
  // Hide all elements of the model
  var ifcApi = new bimdata.IfcApi();
  // Get all doors by filtering with specifying a type
  ifcApi.getElements(cloudId, ifcId, projectId, {type: 'IfcDoor'}).
↪then(function(doors) {
    // Ghost all elements
    viewer.ghost();
    viewer.setUnpickable();

    ....
    viewer.unghost(doorsUuidList);
    viewer.setPickable(doorsUuidList);
    // Color in green doors that are adapted to disabled persons and␣
↪unghost doors
    viewer.setColor(adaptedDoorsUuidList, [0, 1, 0]);
    // Color in red doors that are not adapted to disabled persons and␣
↪unghost doors
    viewer.setColor(unadaptedDoorsUuidList, [1, 0, 0]);
  }, console.error);
});
```

7.5.1.3 Create a new Viewer

You will need to have your accessToken, and the info about the IFC file you want to init the Viewer with: the cloud ID, the project ID, and the IFC ID. These pieces of information are useful to set a new Viewer object in your JS program.

Constructor

The instanciation of the `BIMDataViewer` requires:

- {String}: ID of DOM container element to append the viewer's canvas to, probably a `<div>` element
- **{Object}: Configuration for the Viewer with this attributes as strings:**
  - accessToken
  - cloudId
  - projectId
  - ifcId
- {Boolean} (default = *false*) Whether to make the viewer canvas transparent, which would allow the container to show through.
- {String} (default = *'fr'*) Selects the current localisation language

The code snippet

```
var myAccessToken = 'DEMO_TOKEN';
var myCloudId = 88;
var myProjectId = 100;
var myIfcId = 175;

var viewer = new BIMDataViewer('my-div', {
accessToken: myAccessToken,
cloudId: myCloudId,
projectId: myProjectId,
ifcId: myIfcId
});
```

.... note:: WIP

  This page is not finished yet.

7.5.1.4 How-to get your model into the Viewer

@WIP

Include the Viewer as specified in the *Include viewer recipe*, then, you will use the BIMData API.

Load the Javascript API

To be able to use the API from the same script than your Viewer, include the API by using this line in your HTML file. This will include the `bimdata.ApiClient` you will use to make your API calls.

```
<script type="text/javascript" src="https://unpkg.com/@bimdata/bimdata-
↪api-client/dist/javascript-api-client.min.js"><!-- API call --></
↪script>
```

What info do you will get?

The following informations will be obtained:

- cloud ID, project ID and IFC ID
- the Access Token as a string

Using the API

Create an instance of the API client. You will store it, in this example, in the `defaultClient` var.

```
var defaultClient = bimdata.ApiClient.instance;
```

Then, you will use the API client to get the Bearer object, and you Access Token to get your API Key (see the code):

```
defaultClient.basePath = 'https://api-beta.bimdata.io';
// Configure API key authorization: Bearer
var Bearer = defaultClient.authentications['Bearer'];
Bearer.apiKey = 'Bearer ' + accessToken;
```

Now, you are able to retrieve the Model from your cloud.

7.5.1.5 Parameters and methods

You can change the display of the Viewer using the value of attributes, and using the methods.

Attributes

List of attributes of the `BIMDataViewer`:

- `.helper`: instance of *ViewerHelper* (Type: *object*)
- `.iframe`: the *HTMLElement* <iframe>
- `.elementHovered`: name of the element (Type: *string*), read-only
- `.elementsHovered`: list of names of the elements, read-only
- `.elementHighlighted`: name of the element (Type: *string*)
- `.elementSelected`: name of the element (Type: *string*)
- `.elementsHighlighted`: list of names of the elements (Type: *array*)

- `.elementsSelected`: list of names of the elements (Type: *array*)
- `.elementsHided`: list of names of the elements (Type: *array*)
- `.elementsGhosted`: list of names of the elements (Type: *array*)
- `.buttonsMenuActivated`: { [buttonId]: *boolean* isActivated? } (Type: *Object*), read-only
- `.buttonsMenuShowed`: { [buttonId]: *boolean* isVisible } (Type: *Object*), read-only

Methods

The interface BIMDataViewerInterface let you have actions on the Viewer via many methods.

Delete

`BIMDataViewer.drop()`

Delete the Viewer

Returns void

Window

Methods to interact with custom windows of the Viewer

`BIMDataViewer.addCustomWindow(`*id*, *options*`)`

Creation of a custom window, initialization with the parameters given in the `options` Object

Arguments

- **`id`** (*`string`*) – ID of the Window
- **`options`** (*`object`*) – Instance of BIMViewerParamsCustomWindowOptions

Returns void

```
viewer1.addCustomWindow('first-window', {
    open: true,
    template: 'awesome'
})
```

`BIMDataViewer.addCustomButtonMenu(`*id*, *options*`)`

Attach a button to the Options menu

Arguments

- **`id`** (*`string`*) – ID of the Menu,
- **`options`** (*`object`*) – *Object* BIMViewerCustomButtonOptionsMenu

Returns void

`BIMDataViewer.`**`removeCustomWindow`**`(`*id*`)`

Delete the previously created custom window

Arguments

- **`id`** (*`string`*) – ID of the custom Window

Returns void

`BIMDataViewer.`**`openCustomWindow`**`(`*id*`)`

Display the custom window previously created

Arguments

- **`id`** (*`string`*) – ID of the custom Window

Returns void

`BIMDataViewer.`**`closeCustomWindow`**`(`*id*`)`

Close the custom window previously created

Arguments

- **`id`** (*`string`*) – ID of the custom Window

Returns void

`BIMDataViewer.`**`setCustomWindowTemplate`**`(`*id*, *template*`)`

Apply a template to the custom window

Arguments

- **`id`** (*`string`*) – ID of the custom Window
- **`template`** (*`string`*) – name of the template

Returns void

`BIMDataViewer.`**`onCustomWindow`**`(`*idWindow*, *event*, *selector*, *callback*, *preventDefault*`)`

Associate a behavior, via a function, to the custom window event(s)

Arguments

- **`idWindow`** (*`integer`*) – ID of the custom Window
- **`event`** (*`string`*) – name of the event
- **`selector`** (*`string`*) – CSS-style selector
- **`callback`** (*`function`*) –
- **`preventDefault`** (*`boolean`*) –

Returns integer windowNumber: auto-increment numeration of the custom Windows

`BIMDataViewer.`**`offCustomWindow`**`()`

Event of the disparition of a custom window

Arguments

- **`id`** (*`string`*) – ID of the custom Window

Returns void

Buttons

Methods to interact with the buttons of the Viewer's menus. All these methods return *void*.

`BIMDataViewer.`**`activateButtonMenu`**`(`*target*, *visibility*`)`

Activate a button of the menu

Arguments

- **`target`** (*`string`*) – name of the button
- **`visibility`** (*`boolean`*) –

Returns void

`BIMDataViewer.`**`showButtonMenu`**`(`*target*, *visibility*`)`

Display a button

Arguments

- **`target`** (*`string`*) – name of the button
- **`visibility`** (*`boolean`*) –

Returns void

`BIMDataViewer.`**`showSelectModeMenu`**`(`*target*, *visibility*`)`

:param string target:name of the button :param boolean visibility: :returns: void

`BIMDataViewer.`**`addCustomButtonMenu`**`(`*id*, *options*`)`

Arguments

- **`id`** (*`integer`*) – ID of the menu
- **`options`** (*`object`*) – *Object* instance of `BIMViewerCustomButtonOptionsMenu`

Returns void

`BIMDataViewer.`**`removeCustomButtonMenu`**`(`*id*`)`

Arguments

- **`id`** (*`integer`*) – ID of the menu

Returns void

Reach the Viewer

More generic methods to reach the Viewer and set it.

`BIMDataViewer.`**`on`**`(`*eventName*, *callback*`)`

Associate a behavior, through a function, to the Viewer event(s)

Arguments

- **`eventName`** (*`string`*) – name of the targeted event
- **`callback`** (*`function`*) – *Function* [callback]

Returns number: an auto-increment ID for this Viewer instance

`BIMDataViewer.`**`off`**`(`*id*`)`

Arguments

- **`id`** (*`integer`*) – ID of the Viewer

Returns void

Elements & IFC

Methods to interact with elements of your Model.

`BIMDataViewer.`**`setPickable`**`(`*selector*`)`

Set an element of the model as pickable for selection

Arguments

- **`selector`** (*`string`*) – CSS-style selector

Returns void

`BIMDataViewer.`**`setUnpickable`**`(`*selector*`)`

Set an element of the model as non-pickable for selection

Arguments

- **`selector`** (*`string`*) – CSS-style selector

Returns void

`BIMDataViewer.`**`getElementsInfo`**`(`*ifcId*`)`

Get an element/collection of elements of your model and their information

Arguments

- **`ifcId`** (*`integer`*) –

Returns objects { [id: string]: any }

`BIMDataViewer.`**`getModel`**`(`*uuid*`)`

Get the Model object

Arguments

- **`uuid`** (*`integer`*) –

Returns model

`BIMDataViewer.`**`getStructure`**`(`*uuid*`)`

Get the structure of the Model

Arguments

- **`uuid`** (*`integer`*) –

Returns Promise *Function*

Interface

Methods to modify the display, the view, and point of view.

`BIMDataViewer.`**`getColor`**`(`*id*`)`

Arguments

- **id** (*integer*) – ID of the IFCElement

Returns color: Promise<[*number*, *number*, *number*]

`BIMDataViewer.`**`setColor`**`(`*selector*, *color*`)`

Arguments

- **selector** (*string*) –
- **color** (*array*) – [*number*, *number*, *number*]

Returns void

`BIMDataViewer.`**`getSnapshot`**`(`*options*`)`

Captures a snapshot image of the viewer's canvas.

param object options { *integer* width, *integer* height, *string* format: *"png/jpeg/bmp"* }

param integer width Desired width of result in pixels - defaults to width of canvas.

param integer height Desired height of result in pixels - defaults to height of canvas.

param string format Desired format; "jpeg", "png" or "bmp", default is *"jpeg"*

returns *string* String-encoded image data

```
imageElement.src = viewer.getSnapshot({
    width: 500,
    height: 500,
    format: "png"
});
```

`BIMDataViewer.`**`getViewpoint`**`()`

Returns *Object* instance of <ViewPoint>

`BIMDataViewer.`**`setViewPoint`**`(`*viewpoint*`)`

Arguments

- **viewpoint** (*object*) – instance of <ViewPoint>

Returns void

`BIMDataViewer.`**`viewFit`**`()`

Put the focus on the given element(s)

Arguments

- **selector** (*string*) – CSS-style selector

Returns void

`BIMDataViewer.`**`select`**`()`

Arguments

- **selector** (*string*) – CSS-style selector

Returns void

`BIMDataViewer.`**`deselect`**`()`

Arguments

- **`selector`** (*string*) – CSS-style selector

Returns void

`BIMDataViewer.`**`highlight`**`()`

put the element(s) in the highlight color

Arguments

- **`selector`** (*string*) – CSS-style selector

Returns void

`BIMDataViewer.`**`dehighlight`**`()`

remove the highlight from the element(s)

Arguments

- **`selector`** (*string*) – CSS-style selector

Returns void

`BIMDataViewer.`**`show`**`()`

Arguments

- **`selector`** (*string*) – CSS-style selector

Returns void

`BIMDataViewer.`**`hide`**`()`

Arguments

- **`selector`** (*string*) – CSS-style selector

Returns void

`BIMDataViewer.`**`unghost`**`()`

no more transparency for the given element(s)

Arguments

- **`selector`** (*string*) – CSS-style selector

Returns void

`BIMDataViewer.`**`ghost`**`()`

set transparency to the maximum for the given element(s)

Arguments

- **`selector`** (*string*) – CSS-style selector

Returns void

7.5.1.6 Example: using the viewFit focus

BIMData provides a 3D Viewer with which you can interact with Javascript.

In this example, you want to focus on a given element, and load the Viewer with that focus done. You will need to obtain the UUID of the element you want to focus on, before this example.

The script has 3 parts:

- you get the model from the cloud
- you get the Element you want to set the focus on
- when you load the Viewer, display the Element's properties

See the whole code example on Codepen.io[17].

Get the model in the Viewer

This part is covered by the Recipe "*How-to init the Viewer*" In this recipe, it's assumed that you have the IFC's id.

Get the element you want to set the focus on

To follow best practices of JS developement, you declare the elements and the currentElement variables.

```
var elements = null;
var currentElement = null;
```

For this example, you have previously get the UUID of your element. To get an element, *see the Viewer methods available*.

```
// This is the unique ID of the element whose properties we want to␣
↪display.
var element_uuid = "1XtCE$bZn2XwBDa8za6_lj";
```

Then you write a function that changes the viewFit of the Viewer, making your element the center of the current view. Note that you build the *fullUUID* by concatenating the IFC's id and the UUID itself.

```
/* Selects an object by its UUID, and outputs its properties in the␣
↪console. */
function showObjectByUUID(uuid, showProperties){
  var fullUUID = `${ifcId}#${uuid}`;
  currentElement = elements[fullUUID];
  // Change the element's color
  viewer.setColor(fullUUID, [0.2, 0.5, 0.5]);
  viewer.viewFit(fullUUID);
}
```

[17] https://codepen.io/bimdata/pen/dwpwog

Display the Element's properties

It's now time to use your function. When the Viewer is loading, the viewFit is set to the chosen element.

```
// When the viewer fully loaded the model, display our element's property
viewer.on('viewer-load', function(e) {
    showObjectByUUID(element_uuid, false);
});
```

The complete example

```
var ifcId = 175;

var viewer = new BIMDataViewer('embed', {
  accessToken: 'DEMO_TOKEN',
  cloudId: 88,
  projectId: 100,
  ifcId: ifcId
});

var elements = null;
var currentElement = null;

// This is the unique ID of the element whose properties we want to␣
↪display.
var element_uuid = "1XtCE$bZn2XwBDa8za6_lj";

/* Selects an object by its UUID, and outputs its properties in the␣
↪console. */
function showObjectByUUID(uuid, showProperties){
  var fullUUID = `${ifcId}#${uuid}`;
  currentElement = elements[fullUUID];
  console.log(currentElement);
  // viewer.select(fullUUID);
  // Change the element's color
  viewer.setColor(fullUUID, [0.2, 0.5, 0.5])
  viewer.viewFit(fullUUID)
}

// When the viewer fully loaded the model, display our element's property
viewer.on('viewer-load', function(e) {
elements = viewer.getElementsInfo();
showObjectByUUID(element_uuid, false);
});
```

7.5.1.7 Example: zoom in the Model

8 Playground

8.1 Application

This page is for testing.

8.2 BCF

8.3 Checkplan

8.4 Cloud

8.5 IFC

8.6 Project

8.7 User

8.8 Welcome

The Bimdata API is organized around REST[18]. Our API has predictable, resource-oriented URLs, and uses HTTP response codes to indicate API errors. We use built-in HTTP features, like HTTP authentication and HTTP verbs, which are understood by off-the-shelf HTTP clients. We support cross-origin resource sharing[19], allowing you to interact securely with our API from a client-side web application (though you should never expose your secret API key in any public website's client-side code). JSON is returned by all API responses, including errors, although our API libraries convert responses to appropriate language-specific objects.

[18] https://en.wikipedia.org/wiki/Representational_state_transfer
[19] https://en.wikipedia.org/wiki/Cross-origin_resource_sharing

8.9 Content-Type

Endpoints usually only accept `application/json` content type. Endpoints used for files upload use `application/x-www-form-urlencoded` (aka form-data) content type.

8.10 OpenAPI

Our files are auto-generated from our OpenAPI file[20] with openapi-generator[21].

8.11 Getting Help

8.11.1 Do you need help?

You can reach us by e-mail: support@bimdata.io

[20] https://api-beta.bimdata.io/doc#/
[21] https://github.com/OpenAPITools/openapi-generator

9 Contributing

This documentation is open-source and available on GitHub[22]. Your contributions are welcome. You can either:

- Suggest an edit to this content[23]
- Report any problem with this documentation, please open an issue on GitHub[24].

[22] https://github.com/bimdata/documentation
[23] https://github.com/bimdata/documentation/blob/master/contributing.rst
[24] https://github.com/bimdata/documentation/issues/new

Index