REST Interface - Working with Projects and Entities

Getting all Project Entities

This call retrieves all entities in the project.

Method

GET

URL

<server>/entity.ashx

Parameters

project	Unique identifier for the project

Returns

This call returns a JSON file. The file defines a dictionary where each key is an entity “ID”, and the data associated to the key is another dictionary containing the entity’s properties.

Example

```
http://localhost:58697/entity.ashx?project=myproject
```

Getting an Entity

This call retrieves one entity.

Method

GET

URL

<server>/entity.ashx

Parameters

id	Unique identifier for the entity

Returns

This call returns a JSON file. The file defines a dictionary where the entity’s properties appear as key-value pairs.

Example

http://localhost:58697/entity.ashx?id=770A11061EFA41C0B7729DE5ABA266C4

Requesting a new ID

The REST API provides a mean to obtain a new unique identifier. This is useful when creating object hierarchies where objects are required to point to each other. By generating their IDs first, the application can submit objects only once.

Method

GET

URL

<server>/file.ashx

Parameters

generateids	Contains the number of identifiers to be created

Returns

If completed (200 code), this call returns a JSON array of strings where each string corresponds to a newly created ID:

["21CB001A697043ACBF84F54BCB1244F3"]

Example (PowerShell)

This GET call creates one new unique identifier:

```
Invoke-WebRequest -Uri "http://localhost:58697/file.ashx?generateids=1" -Headers @{
"User-Agent"="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/85.0.4183.102 Safari/537.36"
  "Accept"="*/*"
  "Sec-Fetch-Site"="same-origin"
  "Sec-Fetch-Mode"="cors"
  "Sec-Fetch-Dest"="empty"
  "Referer"="http://localhost:58697/cloud/project.html?id=D9927BF62BFA4CCD930CBC716601D09C"
  "Accept-Encoding"="gzip, deflate, br"
  "Accept-Language"="en-US,en;q=0.9,es;q=0.8"
}
```

Creating a Project

This call creates a new project.

Method

POST

URL

<server>/entity.ashx

Post Payload

Multi-part form data containing two fields:

operation	Must be set to "create"
data	A JSON dictionary where each project property appears as a key-value entry.

Returns

If completed (200 code), this call returns a JSON object that describes the result of the operation and provides the ID of the newly created object:

{"result" : "success", "id" : "D8C328DD186E4059A81555E07E60A210"}

Example (PowerShell)

This POST call creates a folder entity named “New Folder” at the root of the “myproject” project:

```
Invoke-WebRequest -Uri "http://localhost/entity.ashx" -Method "POST" -Headers @{"User-Agent"="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.130 Safari/537.36"; "Accept"="*/*"; "Origin"="http://localhost"; "S
```

Creating an Entity

This call creates a new entity.

Method

POST

URL

<server>/entity.ashx

Parameters

project	Unique identifier for the project

Post Payload

Text payload containing a JSON dictionary where each entity property appears as a key-value entry.

Returns

If completed (200 code), this call returns a JSON object that describes the result of the operation and provides the ID of the newly created object:

{"result" : "success", "id" : "D8C328DD186E4059A81555E07E60A210"}

Example (PowerShell)

This POST call creates a folder entity named “New Folder” at the root of the “myproject” project:

```
Invoke-WebRequest -Uri "http://localhost:58697/entity.ashx?project=myproject" -Method "POST" -Headers @{"Origin"="http://localhost:58697"; "Accept-Encoding"="gzip, deflate, br"; "Accept-Language"="en-US,en;q=0.9,es;q=0.8"; "User-Agent"="custom"; "Accept"="*/*"; "Referer"="http://localhost:58697/cloud/project.html?id=myproject"} -ContentType "application/json; charset=UTF-8" -Body "{`"name`":`"New Folder`",`"type`":`"FILE`",`"file_type`":`"FOLDER`",`"file_date`":`"1557000140020`",`"file_folder`":`"0`"}"
```

Updating an Entity

This call updates the properties of an existing entity.

Method

POST

URL

<server>/entity.ashx

Parameters

id	Unique identifier for the entity
project	Unique identifier for the project

Post Payload

Text payload containing a JSON dictionary where each entity property appears as a key-value entry.

Returns

If completed (200 code), this call returns a JSON object that describes the result of the operation and the ID of the modified object:

{"result" : "success", "id" : " 953F185BDE91400C94B56E0F69F78B16"}

Example (PowerShell)

This POST call renames a folder entity:

Invoke-WebRequest -Uri "http://localhost:58697/entity.ashx?id=953F185BDE91400C94B56E0F69F78B16&project=myproject" -Method "POST" -Headers @{"Origin"="http://localhost:58697"; "Accept-Encoding"="gzip, deflate, br"; "Accept-Language"="en-US,en;q=0.9,es;q=0.8"; "User-Agent"="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 Safari/537.36"; "Accept"="*/*"; "Referer"="http://localhost:58697/cloud/project.html?id=myproject"} -ContentType

"application/json; charset=UTF-8" -Body "{`"name`":`"My Folder`"}"

Deleting Entities

This call deletes a list of entities from the project. If the entity has any data associated to it, it will be deleted as well.

Method

POST

URL

<server>/entity.ashx

Parameters

id	Unique identifier for the entity
project	Unique identifier for the project
org	Organization ID

Post Payload

Multi-part form data containing two fields:

operation	Must be set to “delete”
items	A string containing the IDs for the entities to be deleted, separated by spaces

Returns

If completed (200 code), this call returns a JSON object that describes the result of the deletion operation:

```
{"result" : "success"}
```

Example (PowerShell)

This POST request deletes two entities:

```
Invoke-WebRequest -Uri "http://localhost:58697/entity.ashx?project=myproject&org=2343243456678890" -Method "POST" -Headers @{"Origin"="http://localhost:58697"; "Accept-Encoding"="gzip, deflate, br"; "Accept-Language"="en-US,en;q=0.9,es;q=0.8"; "User-Agent"="custom"; "Accept"="*/*"; "Referer"="http://localhost:58697/cloud/project.html?id=myproject"} -ContentType "multipart/form-data; boundary=----WebKitFormBoundaryn7EOnCC1kkG5mYkA" -Body ([System.Text.Encoding]::UTF8.GetBytes("------WebKitFormBoundaryn7EOnCC1kkG5mYkA$([char]13)$([char]10)Content-Disposition: form-data; name=`"operation`"$([char]13)$([char]10)$([char]13)$([char]10)delete$([char]13)$([char]10)------WebKitFormBoundaryn7EOnCC1kkG5mYkA$([char]13)$([char]10)Content-Disposition: form-data; name=`"items`"$([char]13)$([char]10)$([char]13)$([char]10)953F185BDE91400C94B56E0F69F78B16D8C328DD186E4059A81555E07E60A210 $([char]13)$([char]10)------WebKitFormBoundaryn7EOnCC1kkG5mYkA--$([char]13)$([char]10)"))
```

Getting Entity Files

Some entities may have raw files associated to them. This is the case of raw point clouds and unprocessed block models. Some other entities may have files that are the result of processing spatial data, like a CSV report file, or datasets that have been exported as images, meshes or point clouds.

A single entity may have multiple files associated with it. Each file will have a unique filename under the entity.

Method

GET

URL

<server>/file.ashx

Parameters

id	Unique identifier for the entity
project	Identifier of the project that contains the entity
org	Organization ID
filename	The name of the file inside the entity
namehint	A base64 string that contains a filename that will be suggested by the browser

Returns

This call returns the contents of the file.

Example

http://localhost:58697/file.ashx?org=2343243456678890&project=myproject&id=08A1E6851DFD46F3B0EC958D1118A823&filename=report.csv&namehint=Vm9sdW1lb2ZTcGhlcmUuY3N2

Uploading Entity Files

Some entities may have raw files associated to them. This is the case of raw point clouds and unprocessed block models. Some other entities may have files that are the result of processing spatial data, like a CSV report file, or datasets that have been exported as images, meshes or point clouds.

A single entity may have multiple files associated with it. Each file will have a unique filename under the entity.

Method

POST

URL

<server>/file.ashx

Parameters

id	Unique identifier for the entity
project	Identifier of the project that contains the entity
org	Organization ID
filename	The name of the file inside the entity

Post Payload

Multi-form binary data containing the files to upload.

Returns

If completed (200 code), this call returns a JSON object that describes the result of the upload operation:

```
{"result" : "success", "size" : "0"}
```

Example

```
Invoke-WebRequest -Uri "http://localhost:58697/file.ashx?project=myproject&id=BC094DC8D5CB4D15963D57D13B848A73&org=2343243456678890&partialsize=0" -Method "POST" -Headers @{"Origin"="http://localhost:58697"; "Accept-Encoding"="gzip, deflate, br"; "Accept-Language"="en-US,en;q=0.9,es;q=0.8"; "User-Agent"="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 Safari/537.36"; "Accept"="*/*"; "Referer"="http://localhost:58697/cloud/project.html?id=myproject"} -ContentType "multipart/form-data; boundary=----WebKitFormBoundaryqUAZzU65mbgwqJSz" -Body ([System.Text.Encoding]::UTF8.GetBytes("------WebKitFormBoundaryqUAZzU65mbgwqJSz$([char]13)$([char]10)Content-Disposition: form-data; name=`"file`"; filename=`"Points.zip`"$([char]13)$([char]10)Content-Type: application/x-zip-compressed$([char]13)$([char]10)$([char]13)$([char]10)$([char]13)$([char]10)------WebKitFormBoundaryqUAZzU65mbgwqJSz--$([char]13)$([char]10)"))
```

Uploading large files

In occasions, file sizes can be prohibitively large for uploading using the REST interface, as some services have file limits of 2GB or 4GB for a single file size.

In these cases, there are alternatives to consider:

1. If the files have not been compressed, consider placing them into a ZIP archive before uploading. The processor system will unpack the files.
2. If the entity type allows, consider splitting the spatial data into several files that under the required limit.
3. Bypass the REST upload altogether, and upload directly to the storage system, for instance AWS S3 or Azure Blob Storage.

The following sections cover how to upload data directly, depending on the storage system in use.

Uploading data directly to Azure Blob Storage

In order to upload data directly into Azure Blob storage, it is necessary to know how storage keys are composed in the system.

The keys in Azure are composed using the following syntax:

<OrganizationID>@<ProjectID>@<EntityID>@<FileName>

Where:

OrganizationID - Set to 2343243456678890

ProjectId - The ID of the project that contains the data entity

EntityId - The ID of the entity that will contain the file

FileName - Name for the file

Here is one particular examples of such a key:

2343243456678890@03684CA0F35947EC8243E52972CA3E70@3AC3A444AF414041A0668032536697AD@myblockmodel.zip

This key/blob pair should be created in the Blob Container that is used by the deployment.

Uploading data directly to AWS S3

In order to upload data directly into AWS S3, it is necessary to know how storage keys are composed in the system.

The keys in Azure are composed using the following syntax:

<OrganizationID>@<ProjectID>@<EntityID>@<FileName>

Where:

OrganizationID - Set to 2343243456678890

ProjectId - The ID of the project that contains the data entity

EntityId - The ID of the entity that will contain the file

FileName - Name for the file

Here is one particular examples of such a key:

2343243456678890@03684CA0F35947EC8243E52972CA3E70@3AC3A444AF414041A0668032536697AD@myblockmodel.zip

This key/blob pair should be created in the S3 Bucket that is used by the deployment.

Retrieving Processing Log Files

This call retrieves the processing log for an entity.

Method

GET

URL

<server>/file.ashx

Parameters

id	Unique identifier for the entity
project	Project identifier
org	Organization identifier, use 2343243456678890
filename	Set to processor.log

Returns

This call returns a text file that contains the processing log.

Example

http://localhost:58697/file.ashx?org=2343243456678890&project=18ACF0074ED64B49851033071EA6F401&id=7D5F61152A14495F9BD2216DE59FE778&filename=processor.log

Adding versions to Workflow Products

The process of triggering a new version for a Workflow Product involves the following general steps:

1. Upload all required files in the designated drop-zone blob container
2. Once all required files have successfully uploaded, upload one additional file named "manifest.json" to the designated drop-zone blob container

The "manifest.json" file contains the following properties:

project	Unique identifier for the project
product	The identifier for the Workflow Product
files	A JSON array of strings, where each string contains a path relative the drop-zone blob container for a file that should be attached to the new version
properties	A JSON object containing a dictionary of properties that will be associated with the new version. Values must be strings.

Example of a "manifest.json" file:

```
{

   "project" : "D1847E42C5964B2DAED3B8DBE1C8B6CF",
   "product" : "COM_SURF",
   "files" : ["EOM_220130.zip"],
   "properties": {
      "custom_0" : "value 1",
```

```
      "custom_1" : "value 2"
    }
}
```

This example creates a new version in the COM_SURF product for the specified project. The version will be produced using the "EOM_220130.zip" which must have been previously uploaded to the same blob container as the "manifest.json" file. The example adds two custom properties to the version: "custom_0" and "custom_1".

This is the JSON schema for the "manifest.json" file:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "project": {
      "type": "string"
    },
    "product": {
      "type": "string"
    },
    "files": {
      "type": "array",
      "items": [
        {
          "type": "string"
        }
      ]
    },
    "properties": {
      "type": "object"
    }
  },
  "required": [
    "project",
    "product",
    "files"
  ]
}
```