Python - Data Ingestion

Upload Point Cloud

This example uploads a point cloud dataset.

```
# Import the Voxel Farm Client Library
from voxelfarm import voxelfarmclient

# The URL for the Voxel Farm API
vf_api_url = 'https://vf-api.voxelspace.com'

# Create instance of the Voxel Farm REST API
vf = voxelfarmclient.rest(vf_api_url)

# Get the coordinate system of the project
result = vf.get_project_crs(project)
if not result.success:
    print(result.error_info)
    exit()
crs = result.crs

# Create raw point cloud entity
result = vf.create_entity_raw(
    project=project,
    type=vf.entity_type.RawPointCloud,
    name="My Point Cloud",
    fields={},
    crs=crs)
if not result.success:
    print(result.error_info)
    exit()
pc_id = result.id
print('Raw Point Cloud Entity created ' + pc_id)

# Upload files
files = {'file': open('./mockdata/pointcloud.laz', 'rb')}
result = vf.attach_files(
    project=project,
```

```
    id=pc_id,
    files=files)
if not result.success:
    print(result.error_info)
    exit()
print('Point cloud files uploaded.')
```

Create Indexed Point Cloud

This example triggers the processing of a raw point cloud. This creates a new Indexed Point Cloud.

```
# Create a processed point cloud
# The variable "pc_id" contains the ID of a raw point cloud
# The variable "crs" contains the CRS of the project

result = vf.create_entity_processed(
    project=project,
    type=vf.entity_type.IndexedPointCloud,
    name="My Indexed Point Cloud",
    fields={
        'source': pc_id,
    },
    crs=crs)
if not result.success:
    print(result.error_info)
    exit()
idxpc_id = result.id
print('Indexed point cloud created ' + idxpc_id)
```

Create Voxel Terrain from Point Cloud

This example triggers the processing of a raw point cloud into a Voxel Terrain model

```
# Create a voxel terrain model
# The variable "pc_id" contains the ID of a raw point cloud
# The variable "crs" contains the CRS of the project

result = vf.create_entity_processed(
    project=project,
    type=vf.entity_type.VoxelTerrain,
    name="My Terrain",
    fields={
        'source': pc_id,
        'source_type': vf.entity_type.RawPointCloud,
    },
    crs=crs)
if not result.success:
    print(result.error_info)
    exit()
terrain_id = result.id
```

Upload Mesh Set

This example uploads a set of meshes. To see how "column_meta" property is composed, and what other metadata should be provided along with the meshes, please check the Extended Voxelized Mesh Metadata topic.

```
# Create raw mesh entity
# The variable "crs" contains the CRS for the project

result = vf.create_entity_raw(
    project=project,
    type=vf.entity_type.RawMesh,
    name="My Meshes",
    fields={
        'column_meta': 'File UID ,Index VALUE 0,Conical VALUE 0,Cubic VALUE 0,Spherical VALUE 0'
    },
    crs=crs)
if not result.success:
    print(result.error_info)
    exit(3)
mesh_id = result.id
print('Raw Mesh Entity created ' + mesh_id)

# Upload files
files = {'file': open('./mockdata/meshset.zip', 'rb')}
result = vf.attach_files(
    project=project,
    id=mesh_id,
    files=files)
if not result.success:
    print(result.error_info)
    exit(4)
print('Mesh files uploaded.')
```

Process Mesh Set

This example processes a set of meshes into a voxelized mesh set.

```
# Create a voxel mesh
# The variable "mesh_id" contains the ID of a raw point cloud
# The variable "crs" contains the CRS of the project

result = vf.create_entity_processed(
    project=project,
    type=vf.entity_type.VoxelMesh,
    name="My Mesh Set",
    fields={
        'source': mesh_id
    },
    crs=crs)
if not result.success:
    print(result.error_info)
    exit(5)
voxmesh_id = result.id
```

Create Voxel Terrain from Mesh

This example processes a raw mesh into a Voxel Terrain entity.

```
# The "raw_surface" variable contains the ID of the raw mesh entity
# The "crs" variable contains the project's CRS

result = vf.create_entity_processed(
    project=project,
    type=vf.entity_type.VoxelTerrain,
    name='Terrain Model',
    fields={
        'source': raw_surface,
        'source_type': vf.entity_type.RawMesh
    },
    crs=crs)
if not result.success:
    print(result.error_info)
    exit(5)
surface_id = result.id
```

Upload Block Model

This example uploads a raw Block Model. To see what other metadata should be provided along with the block model files, please check the Extended Block Model Metadata topic.

```
# Create raw mesh entity
# The variable "crs" contains the CRS for the project

result = vf.create_entity_raw(
    project=project,
    type=vf.entity_type.RawBlockModel,
    name="Raw BlockModel",
    fields={},
    crs=crs)
if not result.success:
    print(result.error_info)
    exit(3)
rawbm_id = result.id

# upload metadata
files = {'file': open('./mockdata/process.meta', 'rb')}
result = vf.attach_files(
    project=project,
    id=rawbm_id,
    files=files)
if not result.success:
    print(result.error_info)
    exit(4)
```

Process Block Model

The following example processes a Block Model.

```
# Create a block model
# The variable "rawbm_id" contains the ID of a raw Block Model
# The variable "crs" contains the CRS of the project

result = vf.create_entity_processed(
    project=project,
    type=vf.entity_type.BlockModel,
    name="Block Model",
    fields={
        'source': rawbm_id,
    },
    crs=crs)
if not result.success:
    print(result.error_info)
    exit(5)
bm_id = result.id
```