
geoh5py Documentation

Release 0.6.0-alpha

MiraGeoscience

Jan 27, 2023

CONTENTS

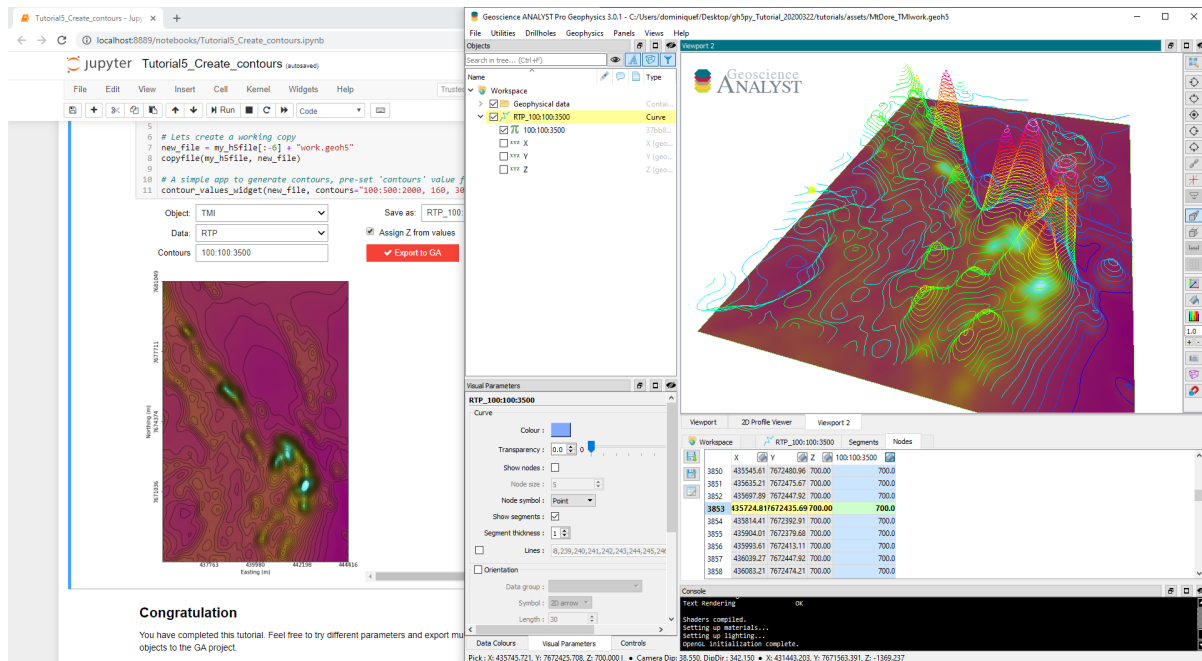
1	In short	3
2	Contents:	5
2.1	Installation	5
2.2	Tutorials	6
2.3	geoh5py	46
2.4	GEOH5 Format	111
2.5	UI.JSON Format	160
2.6	Release Notes	172
2.7	Feedback	174
	Python Module Index	175
	Index	177

Welcome to the documentation page for **geoh5py**!

CHAPTER ONE

IN SHORT

The **geoh5py** library has been created for the manipulation and storage of a wide range of geoscientific data (points, curve, surface, 2D and 3D grids) in **geoh5 file format**. Users will be able to directly leverage the powerful visualization capabilities of **Geoscience ANALYST** along with open-source code from the Python ecosystem.

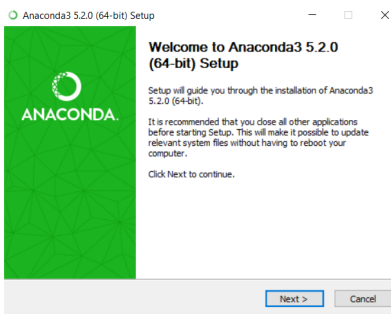


CONTENTS:

2.1 Installation

geoh5py is currently written for Python 3.7 or higher, and depends on **NumPy** and **h5py**.

Note: Users will likely want to take advantage of other packages available in the Python ecosystem. We therefore recommend using **Anaconda** to manage the installation.



Install **geoh5py** from PyPI:

```
$ pip install geoh5py
```

To install the latest development version of **geoh5py**, you can use **pip** with the latest GitHub development branch:

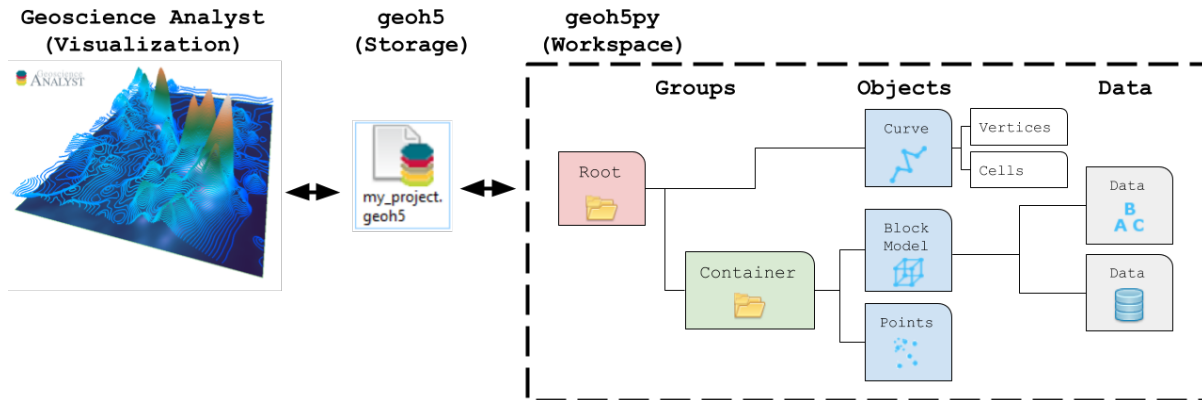
```
$ pip install git+https://github.com/MiraGeoscience/geoh5py.git
```

To work with **geoh5py** source code in development, install from GitHub:

```
$ git clone --recursive https://github.com/MiraGeoscience/geoh5py.git
$ cd geoh5py
$ python setup.py install
```

2.2 Tutorials

This section provides information on how to use the **geoh5py** package, from the creation of a *Workspace* to the creation and manipulation of *Entities*



2.2.1 Workspace

The core element of a project is the *Workspace*. A project *Workspace* holds core information about the author, version and all entities stored in the geoh5 file. It also knows how to create the core structure needed by *Geoscience ANALYST* for visualization.



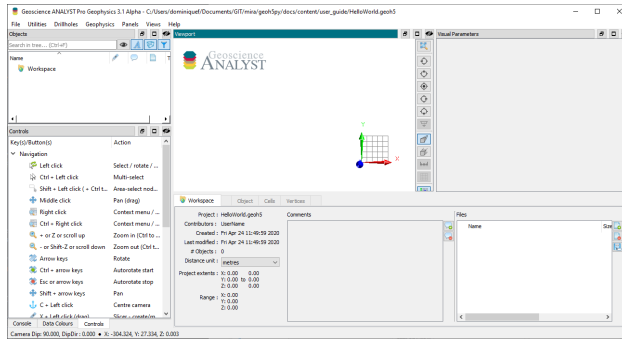
Open and close

You can either open an existing project or create a new project by simply entering the desired file name.

```
[1]: from geoh5py.workspace import Workspace

# Create a new project
workspace = Workspace("my_project.geoh5")
```

Et voila!



By default, the *.geoh5 file is accessed in “read-write” mode. In the eventuality that the file is already used by Geoscience ANALYST, the mode gets changed to “read-only”. This prevents users from modifying the file while used in an active session, but still allows them to extract data from the workspace. The same restriction does not apply to multiple python processes, as permitted by the Single Writer Multiple Reader (SWMR) feature of HDF5.

```
[2]: print(workspace.geoh5)
```

```
<HDF5 file "my_project.geoh5" (mode r+)>
```

After completing the read/write process, the workspace must be closed in order to release the file. Geoscience ANALYST does not allow reading on an opened file.

```
[3]: workspace.close()
```

```
print(workspace._geoh5)
```

```
None
```

Context manager

Likewise, a workspace can be accessed via a context manager which will handle closing the file at the end of a process.

```
[4]: with Workspace("my_project.geoh5") as workspace:
    print(workspace.geoh5)
```

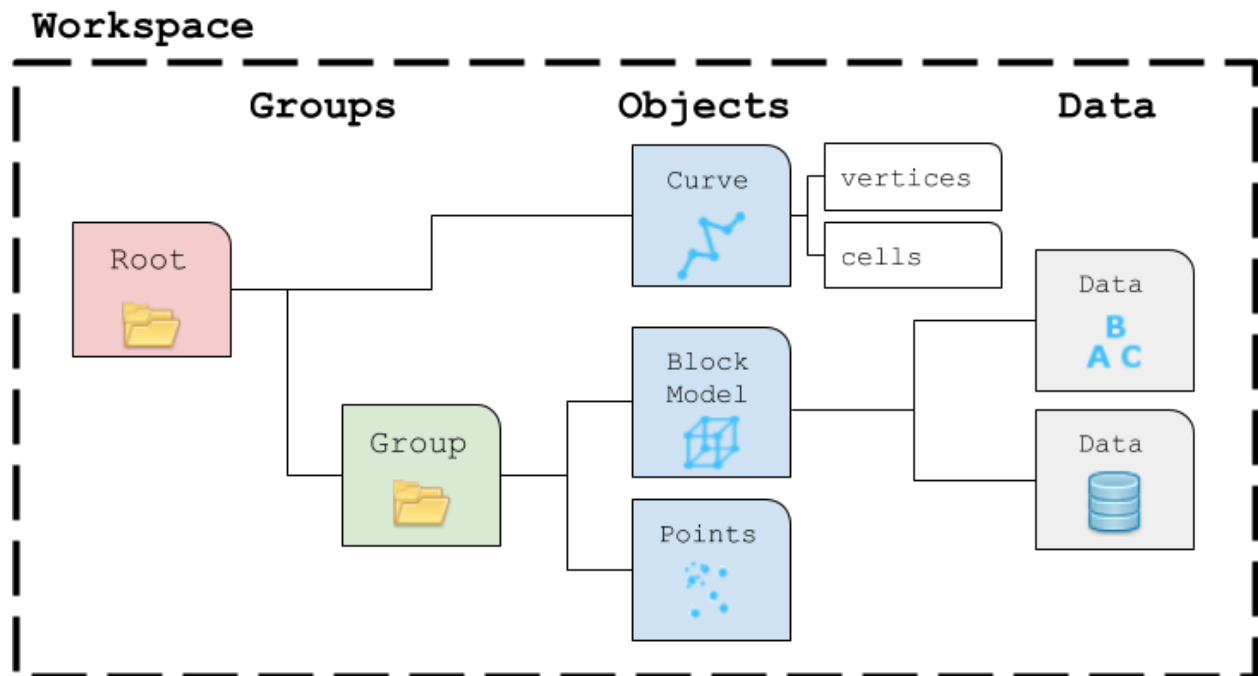
```
print(workspace._geoh5)
```

```
<HDF5 file "my_project.geoh5" (mode r+)>
```

```
None
```

2.2.2 Entities

This section introduces the different entities that can be created and stored in the geoh5 file format.

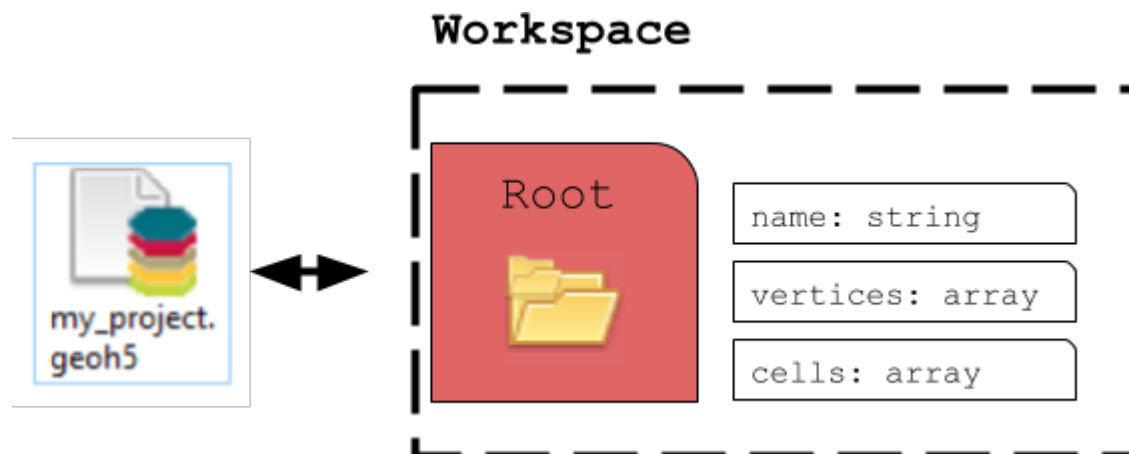


Groups

Groups are effectively containers for other entities, such as Objects (Points, Curve, Surface, etc.) and other Groups. Groups are used to establish parent-child relationships and to store information about a collection of entities.

RootGroup

By default, the parent of any new Entity is the workspace RootGroup. It is the only entity in the Workspace without a parent. Users rarely have to interact with the Root group as it is mainly used to maintain the overall project hierarchy.



ContainerGroup

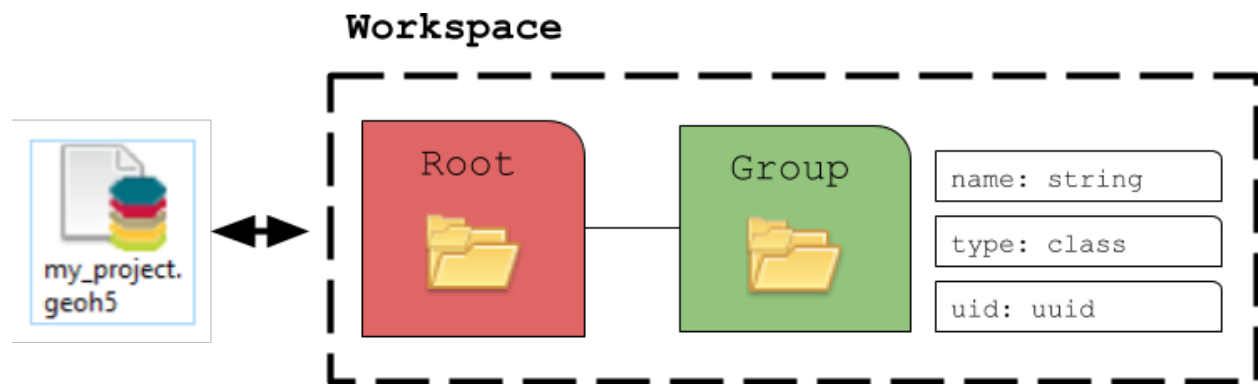
A ContainerGroup can easily be added to the workspace and can be assigned a name and description.

```
[1]: from geoh5py.groups import ContainerGroup
      from geoh5py.workspace import Workspace
      import numpy as np

      # Create a blank project
      workspace = Workspace("my_project.geoh5")

      # Add a group
      group = ContainerGroup.create(workspace, name='myGroup')
```

At creation, "myGroup" is written to the project geoh5 file and visible in the Analyst project tree.



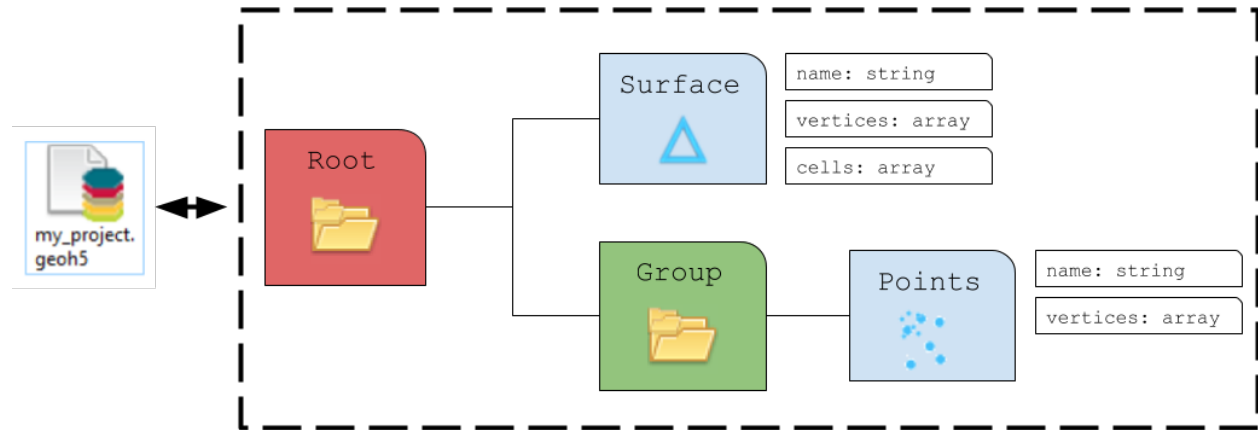
Any entity can be accessed by its name or uid (unique identifier):

```
[2]: print(group.uid)
      print(workspace.get_entity("myGroup")[0] == workspace.get_entity(group.uid)[0])

39d0ccda-b5fe-49e7-b68b-b159f1d5f6a0
True
```

Objects

The geoh5 format enables storing a wide variety of Object entities that can be displayed in 3D. This section describes the collection of Objects entities currently supported by geoh5py.



Points

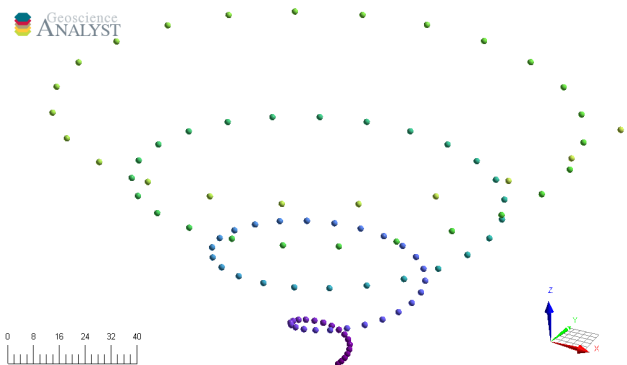
The Points object consists of a list of vertices that define the location of actual data in 3D space. As for all other Objects, it can be created from an array of 3D coordinates and added to any group as follow:

```
[3]: from geoh5py.objects import Points

# Generate a numpy array of xyz locations
n = 100
radius, theta = np.arange(n), np.linspace(0, np.pi*8, n)

x, y = radius * np.cos(theta), radius * np.sin(theta)
z = (x**2. + y**2. )**0.5
xyz = np.c_[x.ravel(), y.ravel(), z.ravel()] # Form a 2D array

# Create the Point object
points = Points.create(
    workspace,      # The target Workspace
    vertices=xyz     # Set vertices
)
```



Curve

The Curve object, also known as a polyline, is often used to define contours, survey lines or geological contacts. It is a sub-class of the Points object with the added `cells` property, that defines the line segments connecting its vertices. By default, all vertices are connected sequentially following the order of the input vertices.

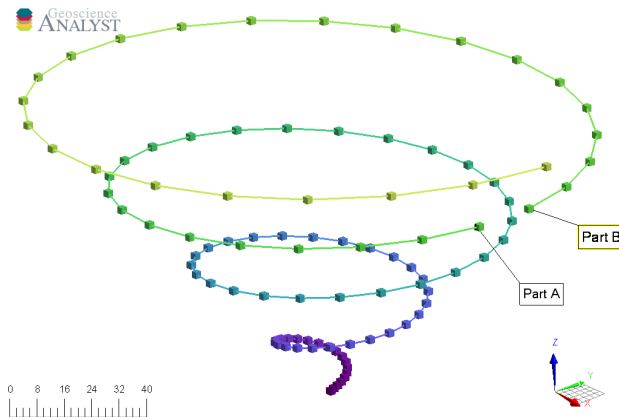
```
[4]: from geoh5py.objects import Curve

# Create the Curve object
curve = Curve.create(
    workspace,      # The target Workspace
    vertices=xyz
)
```

Alternatively, the `cells` property can be modified, either directly or by assigning parts identification to each vertices:

```
[5]: # Split the curve into two parts
part_id = np.ones(n, dtype="int32")
part_id[:75] = 2

# Assign the part
curve.parts = part_id
```



Drillhole

Drillhole objects are different from other objects as their 3D geometry is defined by the collar and surveys attributes. As for version geoh5 v2.0, the drillholes require a DrillholeGroup entity to store the geometry and data.

```
[6]: from geoh5py.groups import DrillholeGroup
from geoh5py.objects import Drillhole

dh_group = DrillholeGroup.create(workspace)

# Create a simple well
total_depth = 100
dist = np.linspace(0, total_depth, 10)
```

(continues on next page)

(continued from previous page)

```

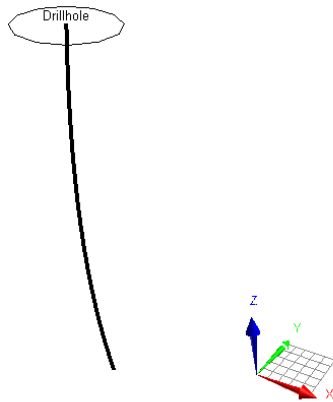
azm = np.ones_like(dist) * 45.
dip = np.linspace(-89, -75, dist.shape[0])
collar = np.r_[0., 10., 10]

well = Drillhole.create(
    workspace, collar=collar, surveys=np.c_[dist, azm, dip], name="Drillhole", parent=dh_
    ↪group
)

print(well.name)

```

Drillhole



Surface

The Surface object is also described by vertices and cells that form a net of triangles. If omitted on creation, the cells property is calculated using a 2D `scipy.spatial.Delaunay` triangulation.

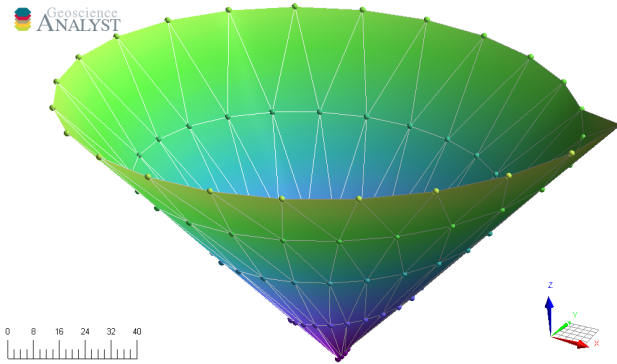
```

[7]: from geoh5py.objects import Surface
     from scipy.spatial import Delaunay

     # Create a triangulated surface from points
     surf_2D = Delaunay(xyz[:, :2])

     # Create the Surface object
     surface = Surface.create(
         workspace,
         vertices=points.vertices, # Add vertices
         cells=surf_2D.simplices
     )

```

GeoImage

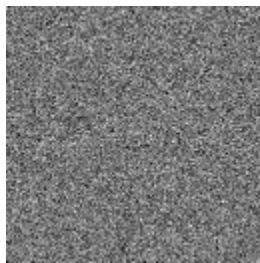
The GeoImage object handles raster data, either single or 3-band images.

```
[8]: from geoh5py.objects import GeoImage

    geoimage = GeoImage.create(workspace)
```

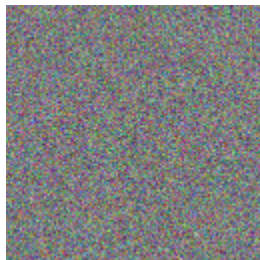
Image values can be assigned to the object from either a 2D numpy.ndarray for single band (gray):

```
[9]: geoimage.image = np.random.randn(128, 128)
    display(geoimage.image)
```



or as 3D numpy.ndarray for 3-band RGB image:

```
[10]: geoimage.image = np.random.randn(128, 128, 3)
    display(geoimage.image)
```

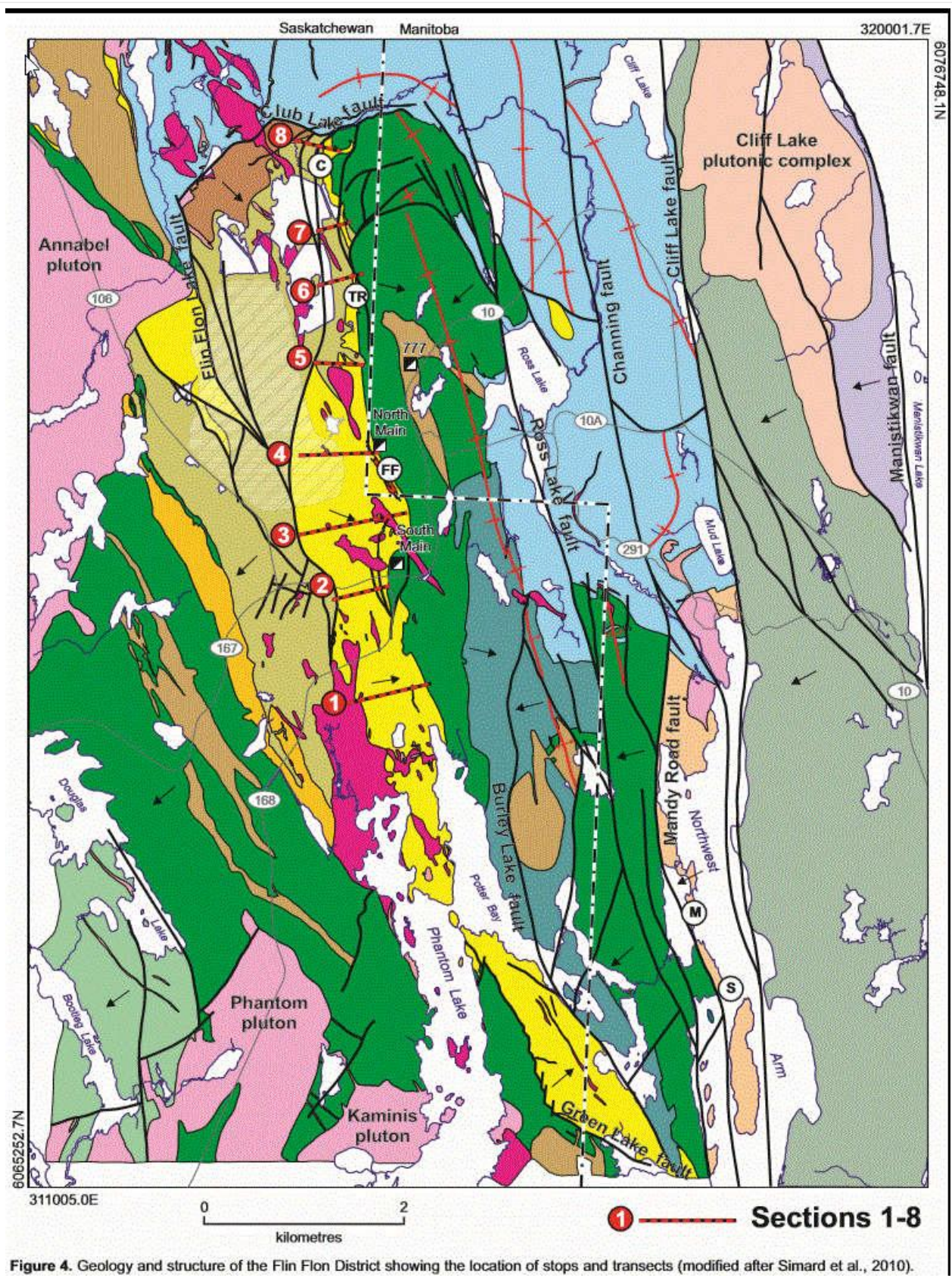


or directly from file (png, jpeg, tiff).

```
[11]: geoimage.image = "./images/flin_flin_geology.jpg"
```

A PIL.Image object gets exposed to the user, which can be used for common raster manipulation (rotation, filtering, etc). The modified raster is stored back on file as a blob (bytes).

```
[12]: display(geoimage.image)
```

Geo-referencing

By default, the `GeoImage` object will be displayed at the origin (xy-plane) with dimensions equal to the pixel count. The utility function `GeoImage.georeference` lets users geo-reference the image in 3D space based on at least three (3) input reference points (pixels) with associated world coordinates.

```
[13]: pixels = [
        [18, 73],
        [757, 1014],
        [18, 1014],
    ]
    coords = [
        [311005, 6065252, 0],
        [320001, 6076748, 0],
        [311005, 6076748, 0]
    ]

    geoimage.georeference(pixels, coords)
    print(geoimage.vertices)

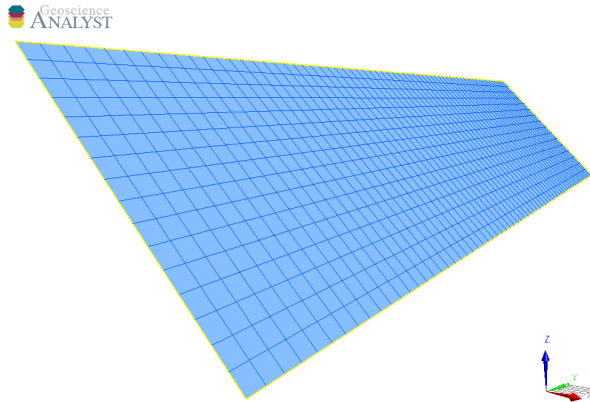
[[ 310785.88227334 6077065.63655685    0.    ]
 [ 320232.29093369 6077065.63655686    0.    ]
 [ 320232.29093369 6064360.17428268    0.    ]
 [ 310785.88227334 6064360.17428268    0.    ]]
```

Grid2D

The `Grid2D` object defines a regular grid of cells often used to display model sections or to compute data derivatives. A `Grid2D` can be oriented in 3D space using the `origin`, `rotation` and `dip` parameters.

```
[14]: from geoh5py.objects import Grid2D

# Create the Surface object
grid = Grid2D.create(
    workspace,
    origin = [25, -75, 50],
    u_cell_size = 2.5,
    v_cell_size = 2.5,
    u_count = 64,
    v_count = 16,
    rotation = 90.0,
    dip = 45.0,
)
```

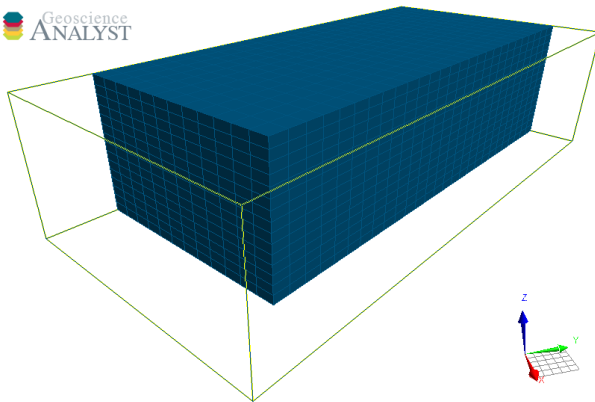



BlockModel

The `BlockModel` object defines a rectilinear grid of cells, also known as a tensor mesh. The cells center position is determined by `cell_delimiters` (offsets) along perpendicular axes (`u`, `v`, `z`) and relative to the origin. `BlockModel` can be oriented horizontally by controlling the `rotation` parameter.

```
[15]: from geoh5py.objects import BlockModel

# Create the Surface object
blockmodel = BlockModel.create(
    workspace,
    origin = [25, -100, 50],
    u_cell_delimiters=np.cumsum(np.ones(16) * 5), # Offsets along u
    v_cell_delimiters=np.cumsum(np.ones(32) * 5), # Offsets along v
    z_cell_delimiters=np.cumsum(np.ones(16) * -2.5), # Offsets along z (down)
    rotation = 30.0
)
```



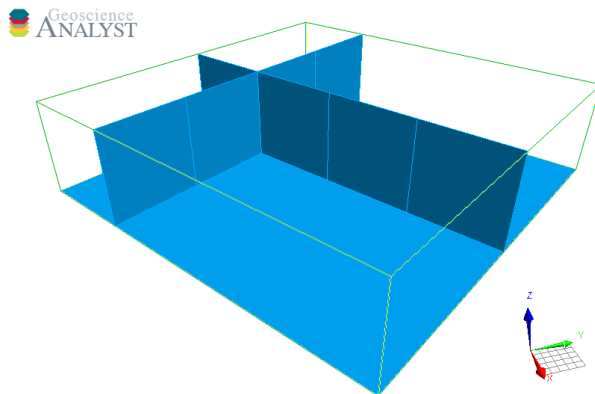
Octree

The Octree object is type of 3D grid that uses a tree structure to define cells. Each cell can be subdivided it into eight octants allowing for a more efficient local refinement of the mesh. The Octree object can also be oriented horizontally by controlling the rotation parameter.

```
[16]: from geoh5py.objects import Octree

octree = Octree.create(
    workspace,
    origin=[25, -100, 50],
    u_count=16,          # Number of cells in power 2
    v_count=32,
    w_count=16,
    u_cell_size=5.0, # Base cell size (highest octree level)
    v_cell_size=5.0,
    w_cell_size=2.5, # Offsets along z (down)
    rotation=30,
)
```

By default, the octree mesh will be refined at the lowest level possible along each axes.

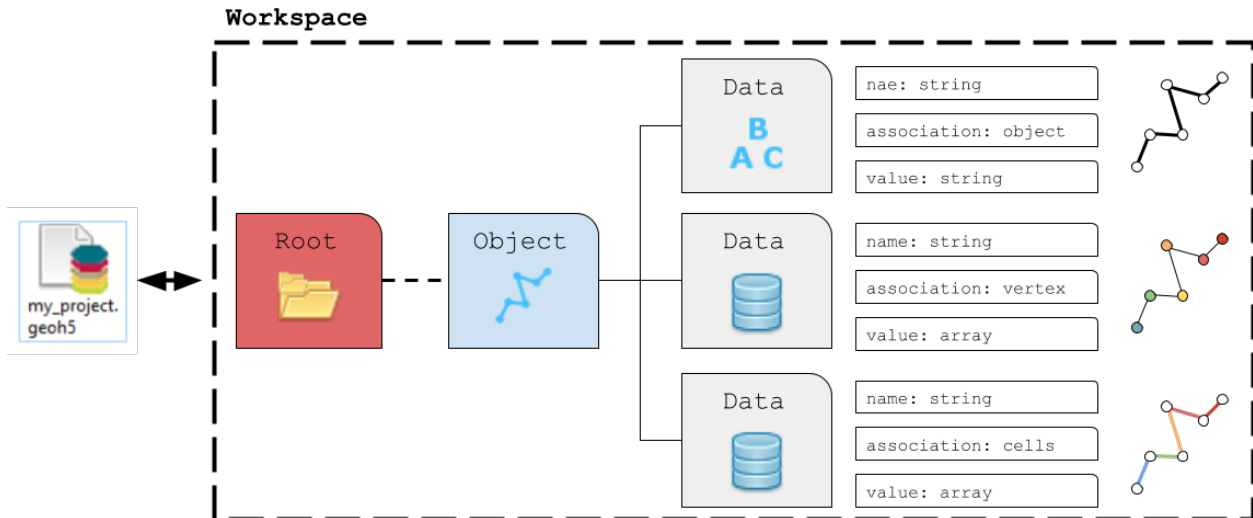


```
[17]: workspace.close()
```

2.2.3 Data

The geoh5 format allows storing data (values) on different parts of an Object. The data types currently supported by geoh5py are

- Float
- Integer
- Text
- Colormap
- Well log



```
[1]: from geoh5py.workspace import Workspace
import numpy as np

# Re-use the previous workspace
workspace = Workspace("my_project.geoh5")

# Get the curve from previous section
curve = workspace.get_entity("Curve")[0]
```

Float

Numerical float data can be attached to the various elements making up object. Data can be added to an Object entity using the `add_data` method.

```
[2]: curve.add_data({
    "my_cell_values": {
        "association": "CELL",
        "values": np.random.randn(curve.n_cells)
    }
})

[2]: <geoh5py.data.float_data.FloatData at 0x7f8cc40643d0>
```

The association can be one of:

- OBJECT: Single element characterizing the parent object
- VERTEX: Array of values associated with the parent object vertices
- CELL: Array of values associated with the parent object cells

The length and order of the array of values must be consistent with the corresponding element of association. If the association argument is omitted, geoh5py will attempt to assign the data to the correct part based on the shape of the data values, either `object.n_values` or `object.n_cells`

```
[3]: # Add multiple data vectors on a single call
data = {}
for ii in range(8):
```

(continues on next page)

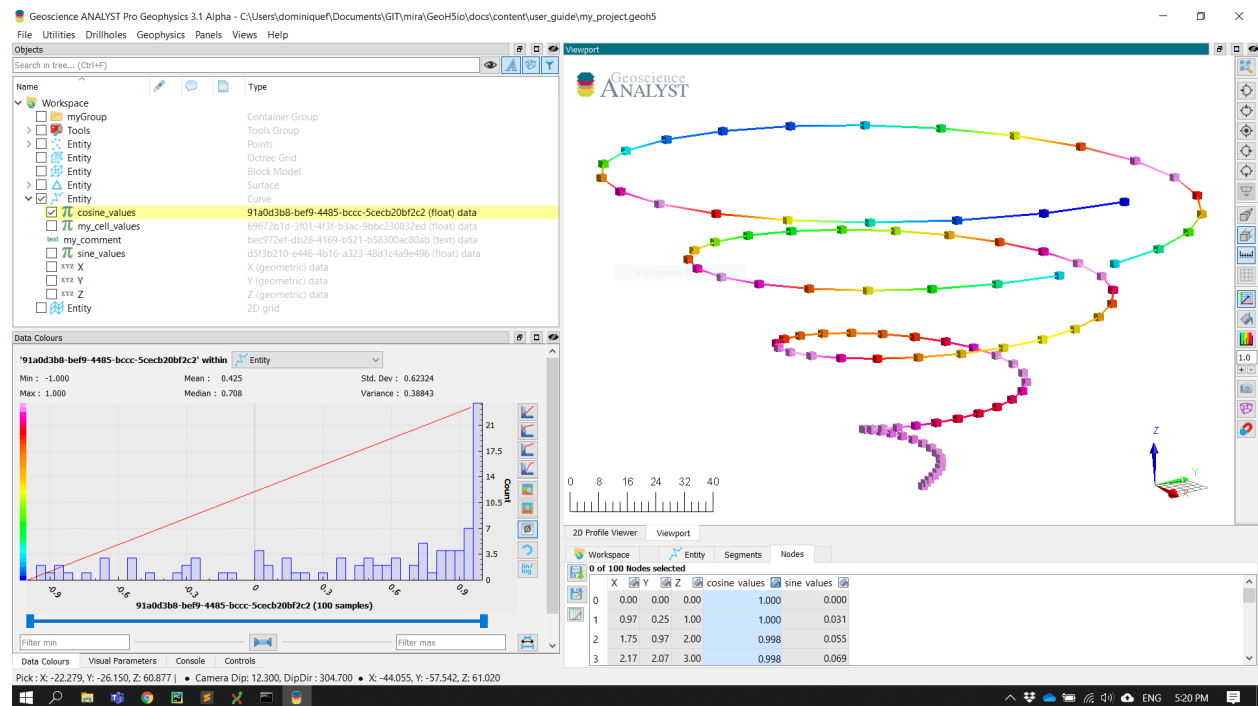
(continued from previous page)

```
data[f"Period:{ii}"] = {
    "association": "VERTEX",
    "values": (ii+1) * np.cos(ii*curve.vertices[:, 0]*np.pi/curve.vertices[:, 0].
    ↪ max()/4.)
}

data_list = curve.add_data(data)
print([obj.name for obj in data_list])

['Period:0', 'Period:1', 'Period:2', 'Period:3', 'Period:4', 'Period:5', 'Period:6',
↪ 'Period:7']
```

The newly created data is directly added to the project's geoh5 file and available for visualization:



Integer

Same implementation as for *Float* data type but with values provided as integer (int32).

Text

Text (string) data can only be associated to the object itself.

```
[4]: curve.add_data({
    "my_comment": {
        "association": "OBJECT",
        "values": "hello_world"
    }
})
```



```
[4]: <geoh5py.data.text_data.TextData at 0x7f8c978d3890>
```

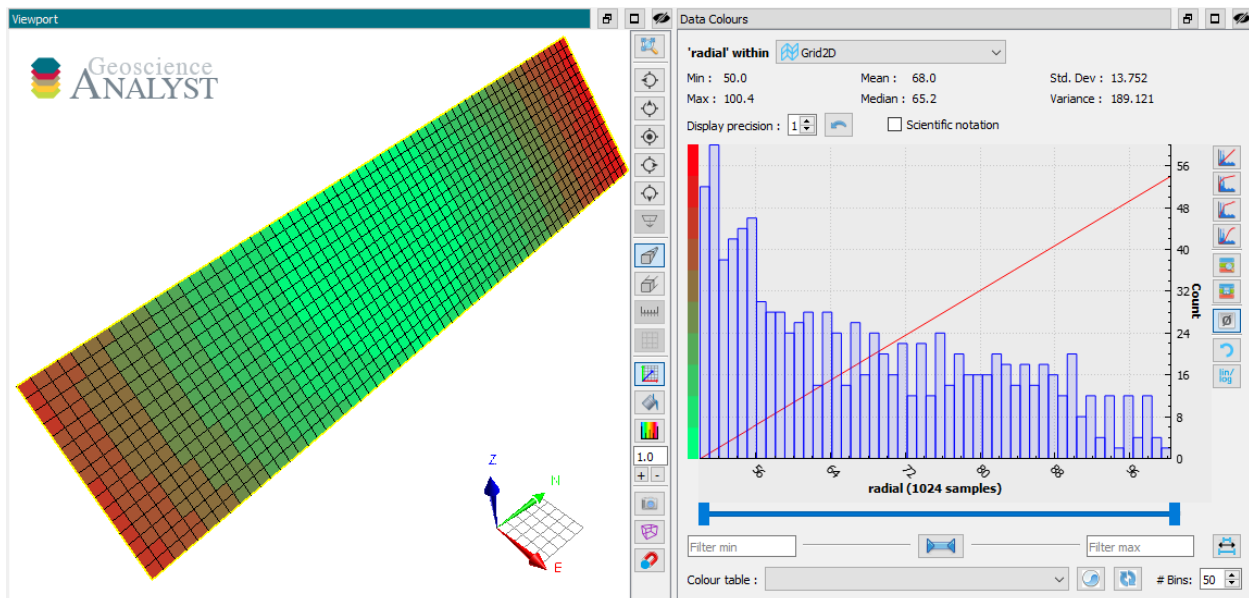
Colormap

The colormap data type can be used to store or customize the color palette used by Geoscience ANALYST.

```
[5]: from geoh5py.data.color_map import ColorMap

# Create some data on a grid2D entity.
grid = workspace.get_entity("Grid2D")[0]

# Add data
radius = grid.add_data({
    "radial": {"values": np.linalg.norm(grid.centroids, axis=1)}
})
```



```
[6]: # Create a simple colormap that spans the data range
nc = 10
rgba = np.vstack([
    np.linspace(radius.values.min(), radius.values.max(), nc), # Values
    np.linspace(0, 255, nc), # Red
    np.linspace(255, 0, nc), # Green
    np.linspace(125, 15, nc), # Blue,
    np.ones(nc) * 255, # Alpha,
]).T
```

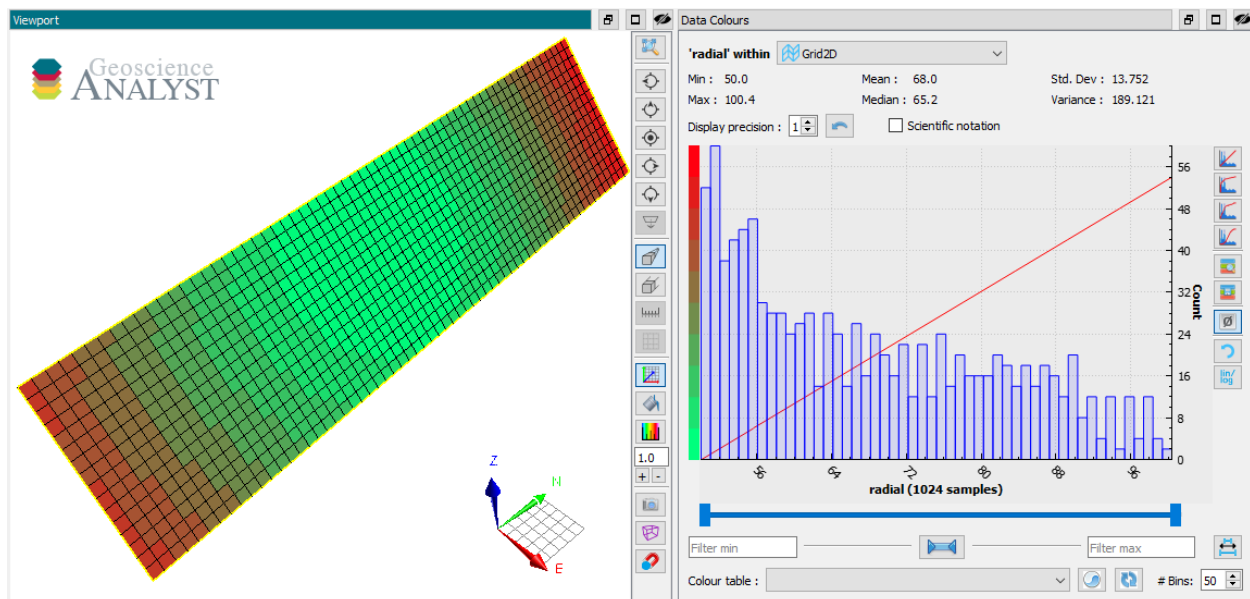
We now have an array that contains a range of integer values for red, green, blue and alpha (RGBA) over the span of the data values. This array can be used to implicitly create a *ColorMap* from the *EntityType*.

```
[7]: # Assign the colormap to the data type
radius.entity_type.color_map = rgba
```

The resulting ColorMap stores the values to geoh5 as a `numpy.recarray` with fields for Value, Red, Green, Blue and Alpha.

```
[8]: radius.entity_type.color_map._values
```

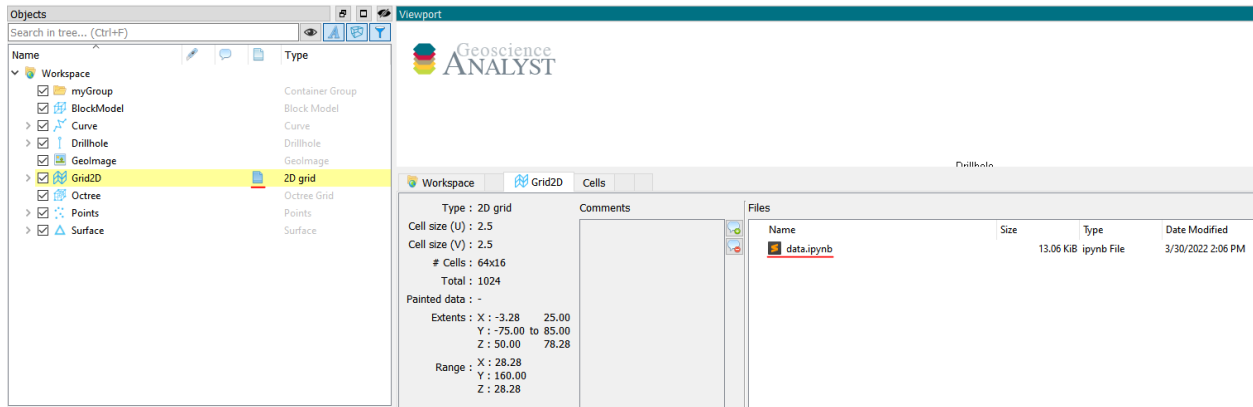
```
[8]: rec.array([( 50.03124024,   0, 255, 125, 255),
 ( 55.62664299,   28, 226, 112, 255),
 ( 61.22204575,   56, 198, 100, 255),
 ( 66.8174485 ,   85, 170,  88, 255),
 ( 72.41285126,  113, 141,  76, 255),
 ( 78.00825401,  141, 113,  63, 255),
 ( 83.60365676,  170,  85,  51, 255),
 ( 89.19905952,  198,  56,  39, 255),
 ( 94.79446227,  226,  28,  27, 255),
 (100.38986503,  255,   0,  15, 255)],
      dtype=[('Value', '<f8'), ('Red', 'u1'), ('Green', 'u1'), ('Blue', 'u1'), ('
↪ Alpha', 'u1')])
```



Files

Raw files can be added to groups and objects and stored as blob (bytes) data in geoh5.

```
[9]: file_data = grid.add_file("./data.ipynb")
```



The information can easily be re-exported out to disk with the save method.

```
[10]: file_data.save_file(path="./temp", name="new_name.ipynb")
```

Well Data

In the case of *Drillhole* objects, data are always stored as from-to interval values.

Depth Data

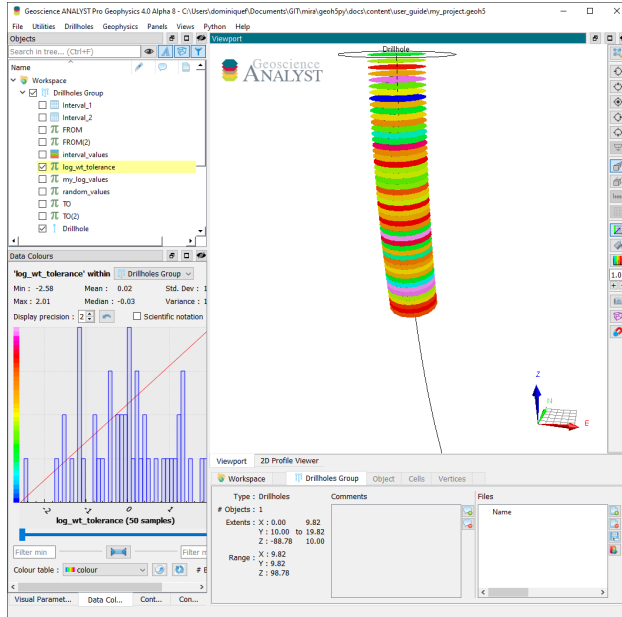
Depth data are used to represent measurements recorded at discrete depths along the well path. A depth attribute is required on creation. Depth markers are converted internally to from-to intervals by adding a small depth values defined by the `collocation_distance`. If the *Drillhole* object already holds depth data at the same location, geoh5py will group the datasets under the same *PropertyGroup*.

```
[12]: well = workspace.get_entity("Drillhole")[0]
depths_A = np.arange(0, 50.) # First list of depth

# Second list slightly offsetted on the first few depths
depths_B = np.arange(0.01, 50.01)

# Add both set of log data with 0.5 m tolerance
well.add_data({
    "my_log_values": {
        "depth": depths_A,
        "values": np.random.randn(depths_A.shape[0]),
    },
    "log_wt_tolerance": {
        "depth": depths_B,
        "values": np.random.randn(depths_B.shape[0]),
    }
})
```

```
[12]: [<abc.FloatDataConcatenated at 0x7f8c958825d0>,
      <abc.FloatDataConcatenated at 0x7f8c958829d0>]
```



Interval (From-To) Data

Interval data are defined by constant values bounded by a start (FROM) and an end (TO) depth. A from-to attribute defined as a `numpy.ndarray (nD, 2)` is expected on creation. Subsequent data are appended to the same interval `PropertyGroup` if the from-to values match within the collocation distance parameter. Users can control the tolerance for matching intervals by supplying a `collocation_distance` argument in meters, or by setting the default on the drillhole entity (`default_collocation_distance = 1e-2` meters).

```
[13]: # Define a from-to array
from_to = np.vstack([
    [0.25, 25.5],
    [30.1, 55.5],
    [56.5, 80.2]
])

# Add some reference data
well.add_data({
    "interval_values": {
        "values": np.asarray([1, 2, 3]),
        "from-to": from_to,
        "value_map": {
            1: "Unit_A",
            2: "Unit_B",
            3: "Unit_C"
        },
    },
    "type": "referenced",
})

# Add float data on the same intervals
well.add_data({
    "random_values": {
```

(continues on next page)

(continued from previous page)

```

    "values": np.random.randn(from_to.shape[0]),
    "from-to": from_to,
}
})

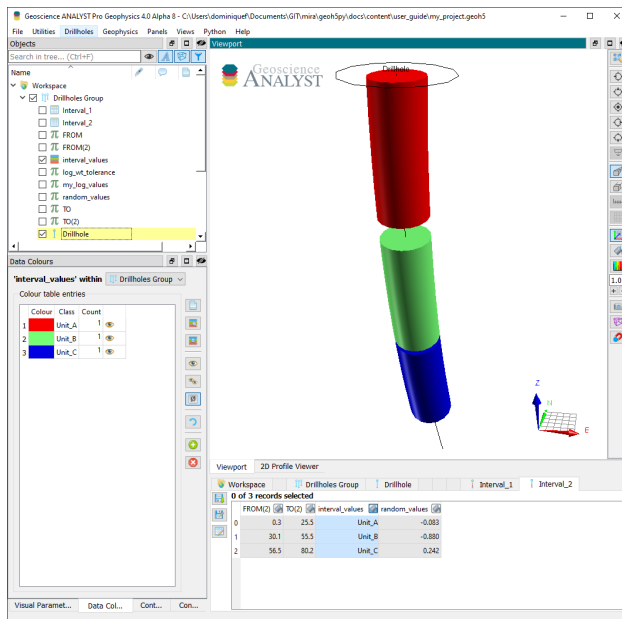
```

```

/home/docs/checkouts/readthedocs.org/user_builds/geoh5py/conda/v0.6.0/lib/python3.7/site-
packages/geoh5py/data/integer_data.py:66: UserWarning: Values provided in int64 are
converted to int32 for PrimitiveTypeEnum.REFERENCED data 'interval_values.'
f"Values provided in {values.dtype} are converted to int32 for "

```

[13]: <abc.FloatDataConcatenated at 0x7f8c978d3a50>



Get data

Just like any Entity, data can be retrieved from the Workspace using the `get_entity` method. For convenience, Objects also have a `get_data_list` and `get_data` method that focusses only on their respective children Data.

```

[14]: my_list = curve.get_data_list()
print(my_list, curve.get_data(my_list[0]))

```

```

['Period:0', 'Period:1', 'Period:2', 'Period:3', 'Period:4', 'Period:5', 'Period:6',
'Period:7', 'my_cell_values', 'my_comment'] [<geoh5py.data.float_data.FloatData object
at 0x7f8cc4046b90>]

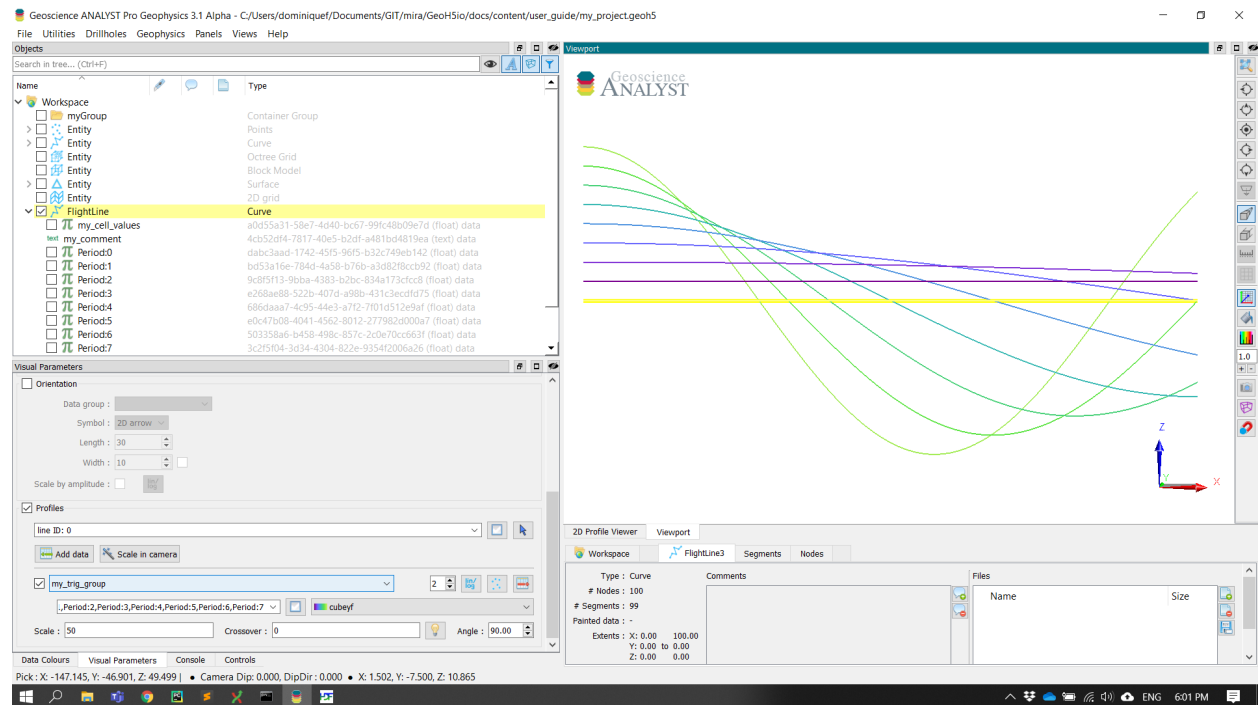
```

2.2.4 Property Groups

Data entities sharing the same parent Object and association can be linked within a `property_groups` and made available through profiling. This can be used to group data that would normally be stored as 2D array.

```
[15]: # Add another VERTEX data and create a group with previous
curve.add_data_to_group([obj.name for obj in data_list], "my_trig_group")
```

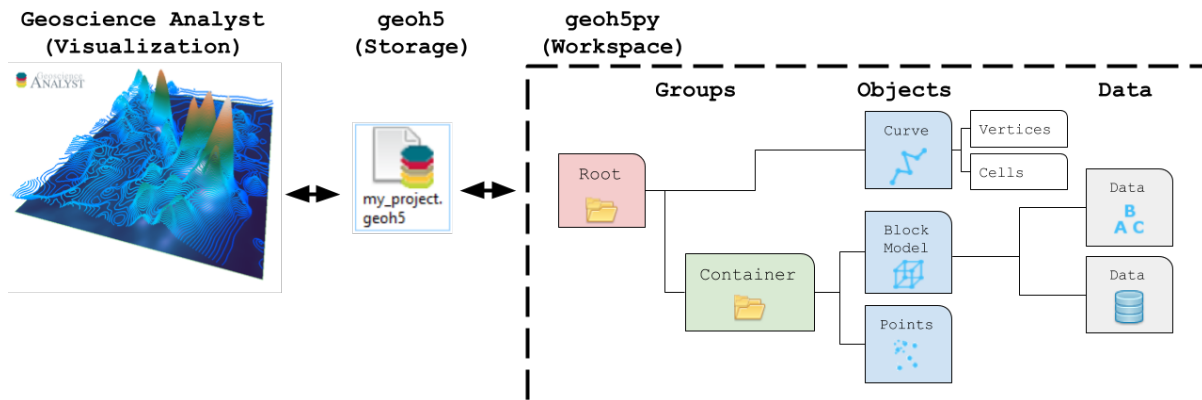
```
[15]: <geoh5py.groups.property_group.PropertyGroup at 0x7f8c9588c410>
```



```
[16]: workspace.close()
```

2.2.5 Surveys

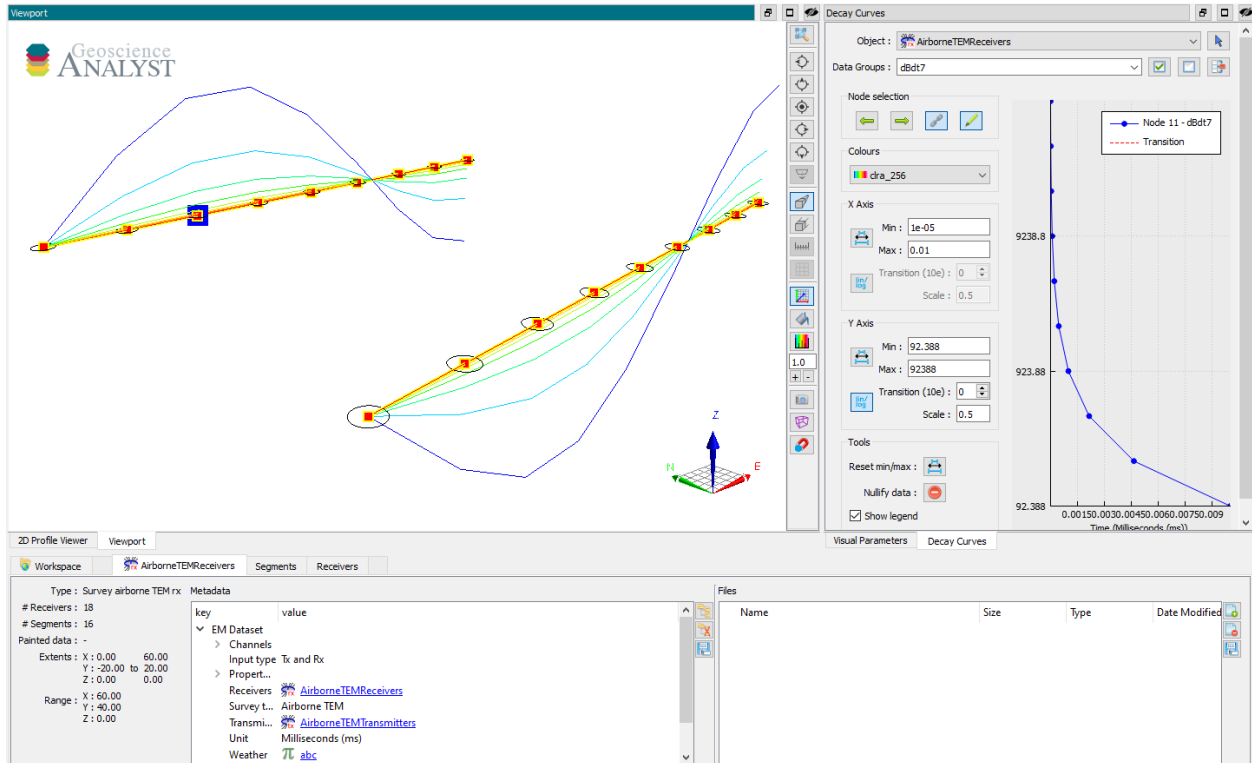
This section provides information on how to create geophysical surveys programmatically.



Airborne Time-Domain

This type of survey can be used to store airborne time-domain electromagnetic (ATEM) data defined by a fixed transmitter-receiver loop configuration. The survey is made up of two entities (*AirborneTEMTransmitters* and *AirborneTEMReceivers*) linked by their metadata.

The following example shows how to generate an airborne TEM survey with associated data stored in geoh5 format and accessible from Geoscience ANALYST.



```
[1]: import numpy as np
from geoh5py.workspace import Workspace
from geoh5py.objects import AirborneTEMReceivers, AirborneTEMTransmitters

# Create a new project
workspace = Workspace("my_project.geoh5")

# Define the pole locations
n_stations = 9
n_lines = 2
x_loc, y_loc = np.meshgrid(np.linspace(0, 60, n_stations), np.linspace(-20, 20., n_lines))
vertices = np.c_[x_loc.ravel(), y_loc.ravel(), np.zeros_like(x_loc).ravel()]

# Assign a line ID to the poles (vertices)
parts = np.kron(np.arange(n_lines), np.ones(n_stations)).astype('int')

# Create the survey as a coincident loop system
aem_receivers = AirborneTEMReceivers.create(workspace, vertices=vertices, parts=parts)
aem_transmitters = AirborneTEMTransmitters.create(workspace, vertices=vertices,
```

(continues on next page)

(continued from previous page)

```
↪parts=parts)
```

We have so far created two separate entities, one for transmitter locations and another for the receivers. In order to finalize the survey, the association must be made between the two entities:

```
[2]: aem_receivers.transmitters = aem_transmitters
```

or equivalently

```
[3]: aem_transmitters.receivers = aem_receivers
```

Only one of the two options above is needed.

Once linked, the two entities will share changes applied to the metadata. For example, changing the `input_type` property on the transmitters yield:

```
[4]: aem_transmitters.input_type = "Tx and Rx"
print(aem_receivers.input_type)
```

```
Tx and Rx
```

Metadata

Along with the survey object itself, the metadata contains all the necessary information to define the geophysical experiment.

```
[5]: aem_receivers.metadata
```

```
[5]: {'EM Dataset': {'Channels': [],
  'Input type': 'Tx and Rx',
  'Property groups': [],
  'Receivers': UUID('913e74ad-6a69-4e89-b06a-601be9fae1e2'),
  'Survey type': 'Airborne TEM',
  'Transmitters': UUID('626e3221-d256-44e7-a257-cbc5fa16147b'),
  'Unit': 'Milliseconds (ms)',
  'Waveform': {'Timing mark': 0.0}}}
```

Channels

List of time channels at which the data are provided.

```
[6]: aem_receivers.channels = np.logspace(-5, -2, 10) # Simple sweep from 1 to 10 ms
```


Input type

Label defining how the survey was created.

- Rx: Survey defined from the `AirborneTEMReceivers` positions, with the `AirborneTEMTransmitters` added from offsets.
- Tx: Survey defined from the `AirborneTEMTransmitters` position, with the `AirborneTEMReceivers` added from offsets.
- Tx and Rx: Survey defined by both the `AirborneTEMTransmitters` and the `AirborneTEMReceivers` positions.

Property groups

List of *PropertyGroups* defining the various data components (e.g. `dBzdt`, `Bz`, ...). It is expected that each component contains data channels at all times and in the same order as defined in `Channels`.

The class method `add_component_data` can help users add data from nested dictionaries. Below is an example using four components:

```
[7]: # Create some simple data
data_fun = lambda t: 1./ t * np.sin(np.pi * (x_loc * y_loc).ravel() / 800.)

# Create a nested dictionary of time data.
data = {
    "dBdt" : {
        f"time[{tt}]": {"values": data_fun(time)} for tt, time in enumerate(aem_
↪receivers.channels)
    }
}

aem_receivers.add_components_data(data)
```

```
[7]: [<geoh5py.groups.property_group.PropertyGroup at 0x7f42c03cab10>]
```

Metadata are also updated to reflect the addition of component data.

```
[8]: aem_receivers.metadata
```

```
[8]: {'EM Dataset': {'Channels': [1e-05,
    2.1544346900318823e-05,
    4.641588833612782e-05,
    0.0001,
    0.00021544346900318823,
    0.00046415888336127773,
    0.001,
    0.002154434690031882,
    0.004641588833612777,
    0.01],
    'Input type': 'Tx and Rx',
    'Property groups': ['dBdt'],
    'Receivers': UUID('913e74ad-6a69-4e89-b06a-601be9fae1e2'),
    'Survey type': 'Airborne TEM',
    'Transmitters': UUID('626e3221-d256-44e7-a257-cbc5fa16147b'),
```

(continues on next page)

(continued from previous page)

```
'Unit': 'Milliseconds (ms)',  
'Waveform': {'Timing mark': 0.0}}}
```

Data channels associated with each component can be quickly accessed through the *BaseEMSurvey.components* property:

```
[9]: aem_receivers.components['dBdt']  
[9]: [<geoh5py.data.float_data.FloatData at 0x7f4296491c50>,  
<geoh5py.data.float_data.FloatData at 0x7f4296484d10>,  
<geoh5py.data.float_data.FloatData at 0x7f4296484d90>,  
<geoh5py.data.float_data.FloatData at 0x7f4296484f50>,  
<geoh5py.data.float_data.FloatData at 0x7f4296484cd0>,  
<geoh5py.data.float_data.FloatData at 0x7f42c03b00d0>,  
<geoh5py.data.float_data.FloatData at 0x7f42c03de110>,  
<geoh5py.data.float_data.FloatData at 0x7f42c03de3d0>,  
<geoh5py.data.float_data.FloatData at 0x7f42c03ca750>,  
<geoh5py.data.float_data.FloatData at 0x7f42c03ca450>]
```

Receivers

Generic label used for surveys to identify the receiver entity. References to itself in the case of AirborneTEMReceivers.

Survey type

Static label identifier for Airborne TEM survey type.

Transmitters

Generic label used for surveys to identify the transmitter entity. References to itself in the case of AirborneTEMTransmitters.

Unit

Units for time sampling of the data - must be one of Seconds (s), Milliseconds (ms), Microseconds (us) or Nanoseconds (ns).

Loop radius

Specifies the transmitter loop radius.

Custom fields

Metadata are stored in geoh5 as a json structure allowing for custom data fields to be added to the survey. Information such as flight data, date/time, offsets, etc. can be added as string, float and int.

```
[10]: aem_receivers.edit_metadata({"Weather": "sunny"})
```

|atem_custom|

Alternatively, a uuid.UUID value can be used if the information is to be provided at every survey position.

```
[11]: # Add a new data entry
abc = aem_receivers.add_data({
    "abc": {"values": np.random.randn(aem_receivers.n_vertices)}
})

# Assign the data as 'Weather' metadata
aem_receivers.edit_metadata({"Weather": abc.uuid})
```

Geoscience ANALYST will automatically create a link referencing the data field to the entity in the project tree.

|atem_uid|

Reserved keywords

For known metadata, such as flight dynamics (yaw, pitch, roll) and offsets (inline, crossline, vertical) the suffix property and value will get replaced based on the input value:

```
[12]: aem_receivers.yaw = 15.
```

|atem_yaw_value|

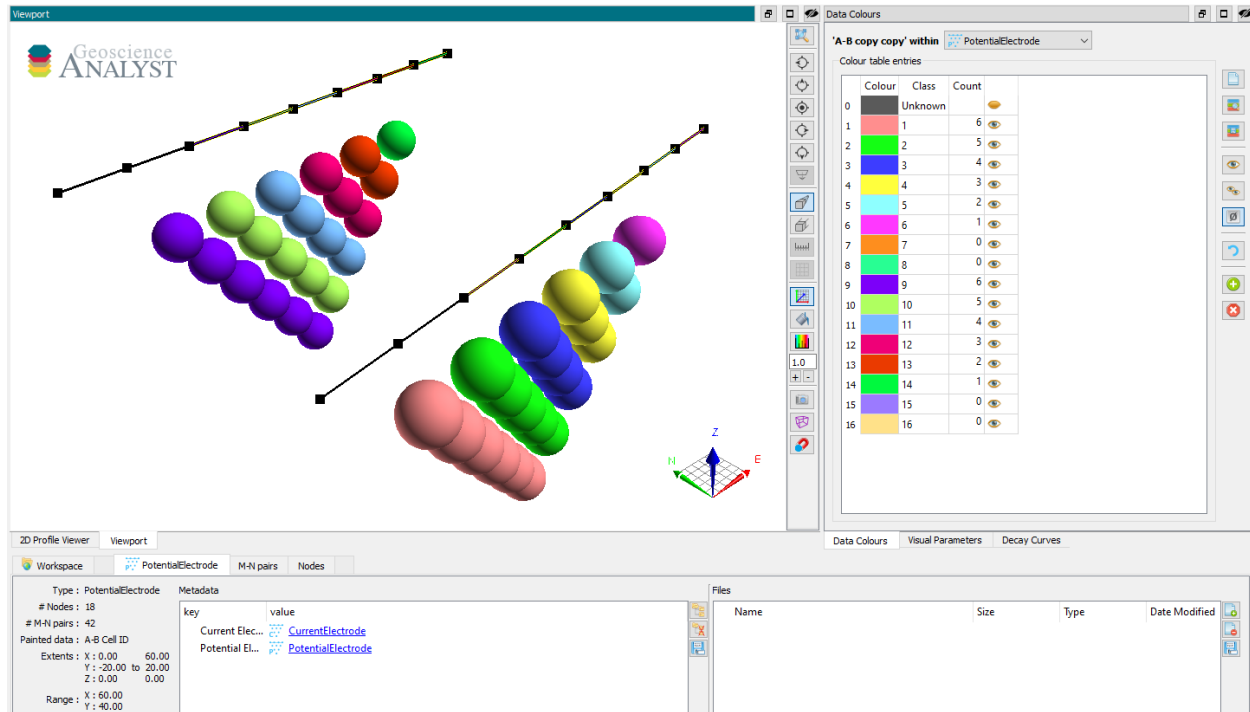
```
[13]: aem_receivers.yaw = abc.uuid # Assign to the yaw property
```

|atem_yaw_property|

Direct Current and Induced Polarization (DC-IP)

This survey type is meant to handle direct-current resistivity data. The survey object is made up of two curve entities defining the transmitter (current) and receiver (potential) electrodes.

The following example shows how to generate a DC-IP survey with associated data stored in geoh5 format and accessible from Geoscience ANALYST.



Current Electrode (transmitters)

The *CurrentElectrode* entity defines the A-B dipole pairs used to inject current into the ground. It is a sub-class of the *PotentialElectrode* object defined by vertices (poles) and cells (dipoles). Here we generate four (4) parallel EW lines with eight dipoles per line.

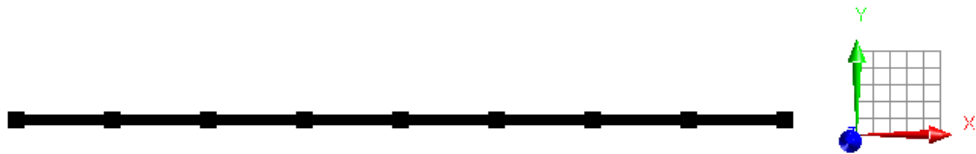
```
[1]: import numpy as np
import uuid
from geoh5py.workspace import Workspace
from geoh5py.objects import CurrentElectrode, PotentialElectrode

# Create a new project
workspace = Workspace("my_project.geoh5")

# Define the pole locations
n_poles = 9
n_lines = 2
x_loc, y_loc = np.meshgrid(np.linspace(0, 60, n_poles), np.linspace(-20, 20., n_lines))
vertices = np.c_[x_loc.ravel(), y_loc.ravel(), np.zeros_like(x_loc).ravel()]

# Assign a line ID to the poles (vertices)
parts = np.kron(np.arange(n_lines), np.ones(n_poles)).astype('int')

# Create the CurrentElectrode object
currents = CurrentElectrode.create(workspace, vertices=vertices, parts=parts)
```



At this stage the `CurrentElectrode` object has segments (cells) connecting all poles in series along line.

AB Cell ID

A key element of the DCIP survey objects is the `ab_cell_id` property. This `ReferenceData` contains the map referencing each cell of the `CurrentElectrode` object to a unique A-B source identifier with name.

The utility function `add_default_ab_cell_id` can help generate this map with a simple name string incrementor.

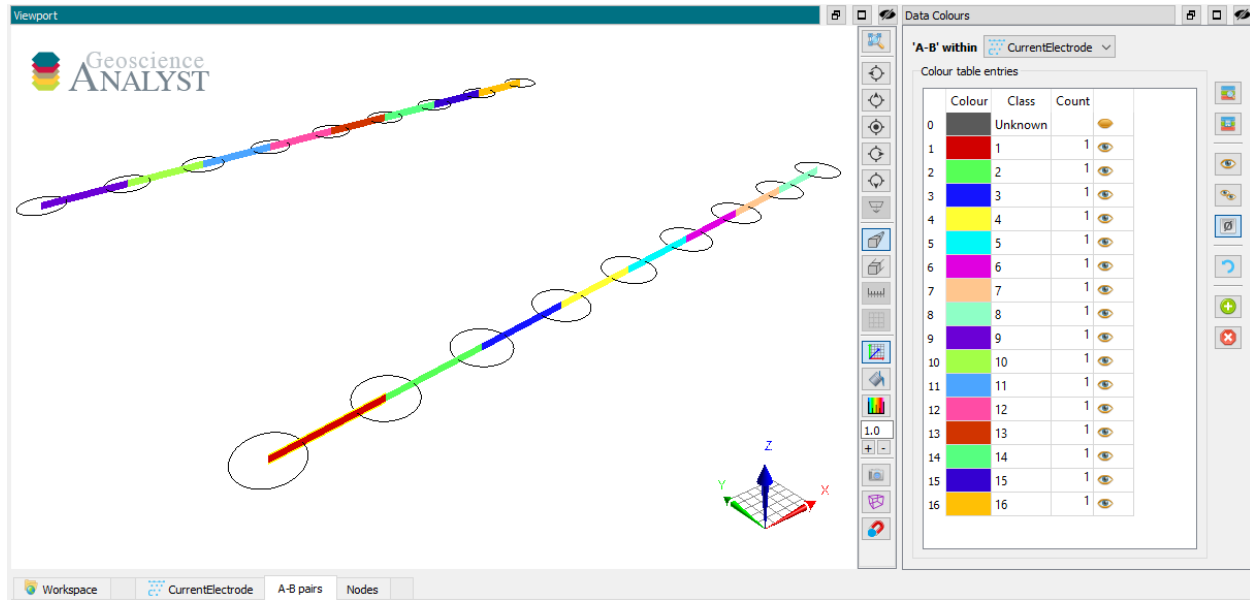
```
[2]: currents.add_default_ab_cell_id()
print(currents.ab_cell_id.value_map.map)

{0: 'Unknown', 1: '1', 2: '2', 3: '3', 4: '4', 5: '5', 6: '6', 7: '7', 8: '8', 9: '9',
 10: '10', 11: '11', 12: '12', 13: '13', 14: '14', 15: '15', 16: '16'}
```

/home/docs/checkouts/readthedocs.org/user_builds/geoh5py/conda/v0.6.0/lib/python3.7/site-packages/geoh5py/data/integer_data.py:66: UserWarning: Values provided in int64 are converted to int32 for PrimitiveTypeEnum.REFERENCED data 'A-B Cell ID.'

f"Values provided in {values.dtype} are converted to int32 for "

In this specific case, every cell on the curve corresponds to a unique dipole source current. For more complex survey configurations, users can edit the `cell` property in order to define different combinations of connections between poles.



Note: The first entry `{0:Unknown}` is a reserved field used by Geoscience ANALYST to flag unknown data entries.

Potential Electrode (receivers)

The *PotentialElectrode* object defines the M-N dipole pairs used to measure the electric potential (receivers). It is a sub-class of the *Curve* object defined by vertices (poles) and cells (dipoles).

Although poles could be set independently on the *CurrentElectrode* and *PotentialElectrode* objects, here we re-uses the same locations for simplicity:

```
[3]: potentials = PotentialElectrode.create(workspace, vertices=vertices)
```

Next, we must define the receiver dipoles. The following routine generates a maximum of six (6) receivers dipoles per injection currents along line.

```
[4]: N = 6
dipoles = []
current_id = []

for val in currents.ab_cell_id.values: # For each source dipole
    if val == 0: # Skip the unknown
        continue

    cell_id = val - 1 # Python 0 indexing
    line = currents.parts[currents.cells[cell_id, 0]]
    for m_n in range(N):
        dipole_ids = (currents.cells[cell_id, :] + 2 + m_n).astype("uint32") # Skip two...
        ↪ poles

        # Shorten the array as we get to the end of the line
        if (
            any(dipole_ids > (potentials.n_vertices - 1))
            or any(currents.parts[dipole_ids] != line)
        ):
            # ...
```

(continues on next page)

(continued from previous page)

continue

```

dipoles += [dipole_ids] # Save the receiver id
current_id += [val] # Save the source id

potentials.cells = np.vstack(dipoles)

```

Finally, users need to create an association between each receiver dipole (M-N) to a dipole current (A-B). The mapping is done through the `ab_cell_id` property of the `PotentialElectrode`. An integer (ID) value must be assigned to each cell, corresponding to the *AB Cell ID* pairs stored on the associated `CurrentElectrode` object.

```
[5]: potentials.ab_cell_id = np.asarray(current_id, dtype="int32")
```

Metadata

The link between the sources *CurrentElectrode* and the receivers *PotentialElectrode* is established by the metadata, shared by both entities. The connection can be made by assigning `current_electrodes` to the receivers:

```
[6]: potentials.current_electrodes = currents
```

or equivalently by setting `potential_electrodes` to the currents

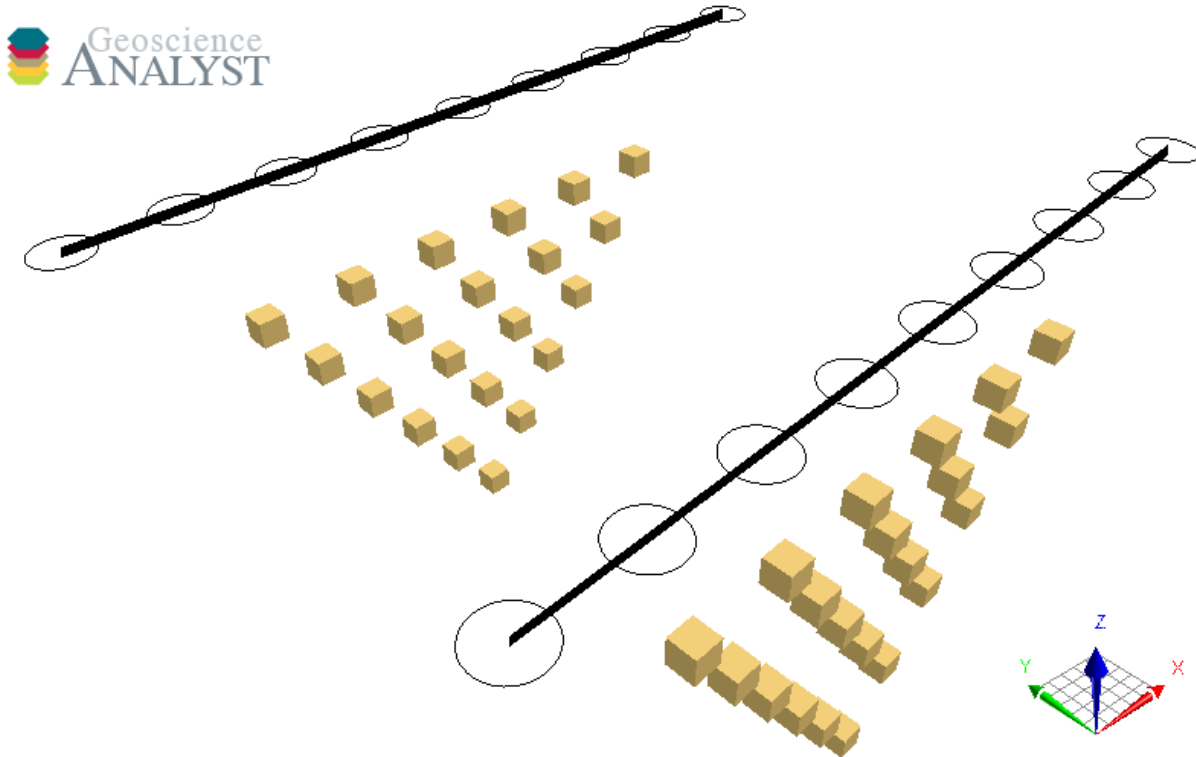
```
[7]: currents.potential_electrodes = potentials
```

In both cases, the link between the two objects gets encoded automatically to their respective metadata.

```
[8]: print(potentials.metadata == currents.metadata)
currents.metadata
```

```
True
```

```
[8]: {'Current Electrodes': UUID('c9cc76a3-f0dc-4db3-8deb-8d14881812ce'),
      'Potential Electrodes': UUID('531672aa-bb0b-44db-a2aa-d331a32f5f1c')}
```



Note: The `ab_cell_id` property of the `CurrentElectrode` and `PotentialElectrode` are two different `ReferenceData` entities:

```
[9]: print(potentials.ab_cell_id == currents.ab_cell_id)
```

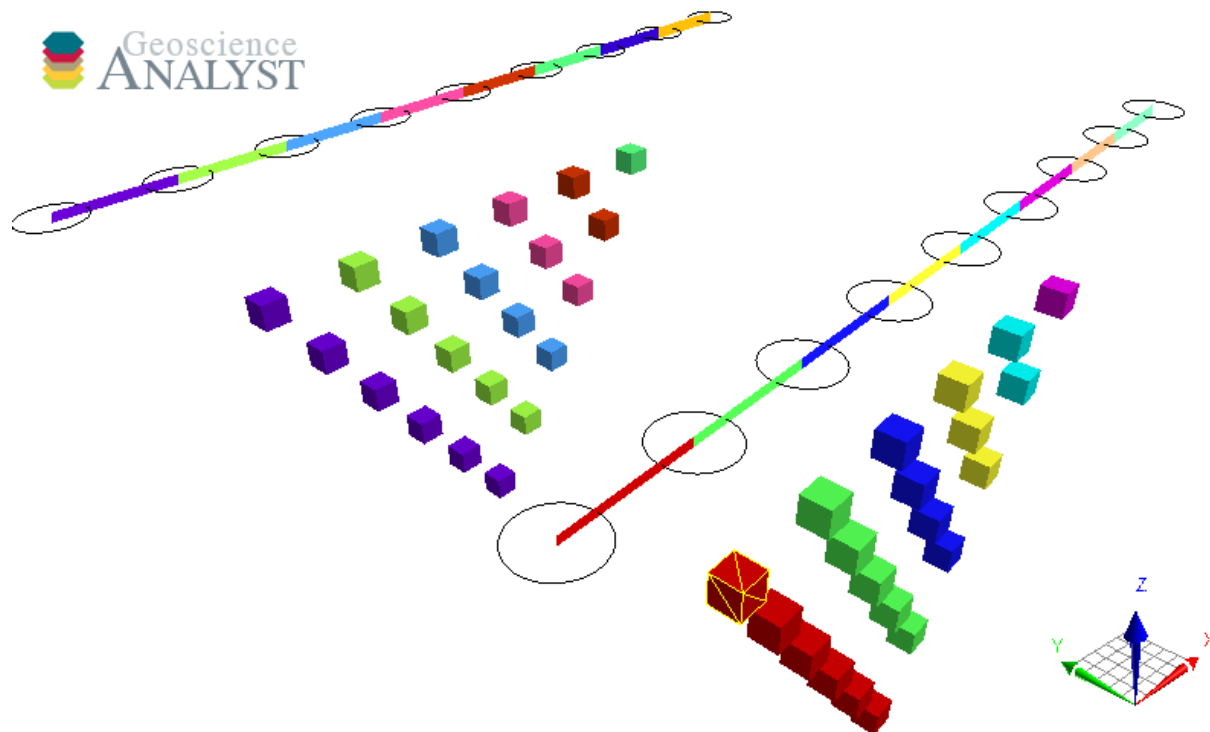
```
False
```

but share the same `DataType` that holds the map of unique source dipoles.

```
[10]: print(potentials.ab_cell_id.entity_type == currents.ab_cell_id.entity_type)
```

```
True
```

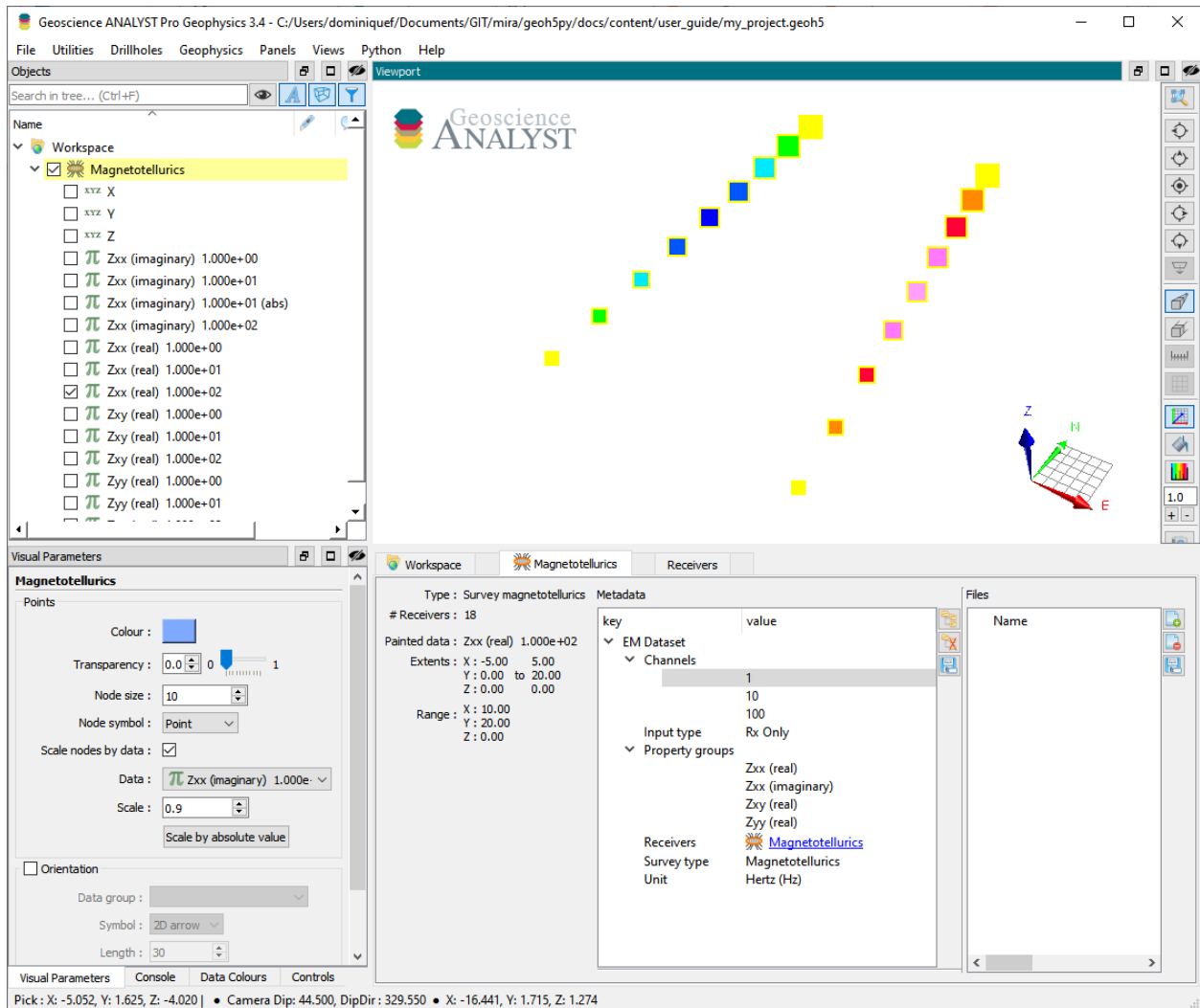
This link between `DataType` allows users to query the data by dipole sources and display the values as pseudo-section in [Geoscience ANALYST](#)



Magnetotellurics

This object can be used to store magnetotelluric (MT) surveys - a natural-source geophysical method. Data are provided in the frequency-domain as point source measurements of either impedances or apparent resistivity/phase.

The following example shows how to generate an MT survey with associated data stored in geoh5 format and accessible from [Geoscience ANALYST](#).



```
[1]: import numpy as np
from geoh5py.workspace import Workspace
from geoh5py.objects import MTReceivers

# Create a new project
workspace = Workspace("my_project.geoh5")

# Define a synthetic survey with receivers on 2 lines, 60 m apart
x_loc, y_loc = np.meshgrid(np.linspace(-5, 5, 2), np.linspace(0., 20., 9))
vertices = np.c_[x_loc.ravel(), y_loc.ravel(), np.zeros_like(x_loc).ravel()]

# Create the survey from vertices
mt_survey = MTReceivers.create(workspace, vertices=vertices)
```

Only receivers are needed to define the survey as MT uses the ambient electromagnetic field of the Earth - no transmitters (source) required.

Metadata

Along with the *MTReceivers*, the metadata contains all the necessary information to define the geophysical experiment.

```
[2]: mt_survey.metadata
[2]: {'EM Dataset': {'Channels': [],
  'Input type': 'Rx only',
  'Property groups': [],
  'Receivers': UUID('5c9cfc28-9d29-4bf4-a239-a62b71b268df'),
  'Survey type': 'Magnetotellurics',
  'Unit': 'Hertz (Hz)'}}
```

Channels

List of frequencies at which the data are provided.

```
[3]: mt_survey.channels = [1., 10., 100.]
```

Input type

Generic label used in the geoh5 standard for all EM survey entities. Restricted to `Rx only` in the case of natural sources methods.

Property groups

List of *PropertyGroups* defining the various data components (e.g. `Zxx (real)`, `Zxy (imag)`, ...). It is not required to supply all components of the impedance tensor, but it is expected that each component contains a list of data channels of length and in the same order as the `Channels` (one Data per frequency).

The class method `add_components_data` can help users add data from nested dictionaries. Below is an example using four components:

```
[4]: # Arbitrary data generator using sine functions
data_fun = lambda c, f: (c+1.) * np.sin(f * np.pi * (x_loc * y_loc).ravel() / 200.)

# Create a nested dictionary of component and frequency data.
data = {
    component : {
        f"{component}_{freq}": {"values": (ff+1)*1000. + (cc+1) * 100. + np.
→ arange(vertices.shape[0])} for ff, freq in enumerate(mt_survey.channels)
    } for cc, component in enumerate([
        "Zxx (real)", "Zxx (imaginary)",
        "Zxy (real)", "Zxy (imaginary)",
        "Zyx (real)", "Zyx (imaginary)",
        "Zyy (real)", "Zyy (imaginary)",
    ])
}

mt_survey.add_components_data(data)
```

```
[4]: [<geoh5py.groups.property_group.PropertyGroup at 0x7f1a5c04c350>,
<geoh5py.groups.property_group.PropertyGroup at 0x7f1a80756c50>,
<geoh5py.groups.property_group.PropertyGroup at 0x7f1a56faad50>,
<geoh5py.groups.property_group.PropertyGroup at 0x7f1a56fb2950>,
<geoh5py.groups.property_group.PropertyGroup at 0x7f1a56fb44d0>,
<geoh5py.groups.property_group.PropertyGroup at 0x7f1a56fb4fd0>,
<geoh5py.groups.property_group.PropertyGroup at 0x7f1a56fb6c50>,
<geoh5py.groups.property_group.PropertyGroup at 0x7f1a56fb87d0>]
```

Metadata are updated immediately to reflect the addition of components:

```
[5]: mt_survey.metadata
```

```
[5]: {'EM Dataset': {'Channels': [1.0, 10.0, 100.0],
'Input type': 'Rx only',
'Property groups': ['Zxx (real)',
'Zxx (imaginary)',
'Zxy (real)',
'Zxy (imaginary)',
'Zyx (real)',
'Zyx (imaginary)',
'Zyy (real)',
'Zyy (imaginary)'],
'Receivers': UUID('5c9cfc28-9d29-4bf4-a239-a62b71b268df'),
'Survey type': 'Magnetotellurics',
'Unit': 'Hertz (Hz)'}]}
```

Data channels associated with each component can be quickly accessed through the [BaseEMSurvey.components](#) property:

```
[6]: mt_survey.components
```

```
[6]: {'Zxx (real)': [<geoh5py.data.float_data.FloatData at 0x7f1a57b4b610>,
<geoh5py.data.float_data.FloatData at 0x7f1a8073d750>,
<geoh5py.data.float_data.FloatData at 0x7f1a8076e290>],
'Zxx (imaginary)': [<geoh5py.data.float_data.FloatData at 0x7f1a56fa6f50>,
<geoh5py.data.float_data.FloatData at 0x7f1a8076ed90>,
<geoh5py.data.float_data.FloatData at 0x7f1a80756f50>],
'Zxy (real)': [<geoh5py.data.float_data.FloatData at 0x7f1a807568d0>,
<geoh5py.data.float_data.FloatData at 0x7f1a56faa850>,
<geoh5py.data.float_data.FloatData at 0x7f1a56faad90>],
'Zxy (imaginary)': [<geoh5py.data.float_data.FloatData at 0x7f1a56faafd0>,
<geoh5py.data.float_data.FloatData at 0x7f1a56fb2790>,
<geoh5py.data.float_data.FloatData at 0x7f1a56fb28d0>],
'Zyx (real)': [<geoh5py.data.float_data.FloatData at 0x7f1a56fb2b50>,
<geoh5py.data.float_data.FloatData at 0x7f1a56fb4310>,
<geoh5py.data.float_data.FloatData at 0x7f1a56fb4450>],
'Zyx (imaginary)': [<geoh5py.data.float_data.FloatData at 0x7f1a56fb4410>,
<geoh5py.data.float_data.FloatData at 0x7f1a56fb4e10>,
<geoh5py.data.float_data.FloatData at 0x7f1a56fb4f90>],
'Zyy (real)': [<geoh5py.data.float_data.FloatData at 0x7f1a56fb61d0>,
<geoh5py.data.float_data.FloatData at 0x7f1a56fb6b50>,
<geoh5py.data.float_data.FloatData at 0x7f1a56fb6b10>],
'Zyy (imaginary)': [<geoh5py.data.float_data.FloatData at 0x7f1a56fb68d0>,
<geoh5py.data.float_data.FloatData at 0x7f1a56fb6850>,
<geoh5py.data.float_data.FloatData at 0x7f1a56fb6810>]}
```

(continues on next page)

(continued from previous page)

```
<geoh5py.data.float_data.FloatData at 0x7f1a56fb8550>,  
<geoh5py.data.float_data.FloatData at 0x7f1a56fb8690>]]}
```

Receivers

Generic label used in the geoh5 standard for EM survey to identify the receiver entity. Restricted to itself in the case of MTReivers.

Survey type

Label identifier for Magnetotellurics survey type.

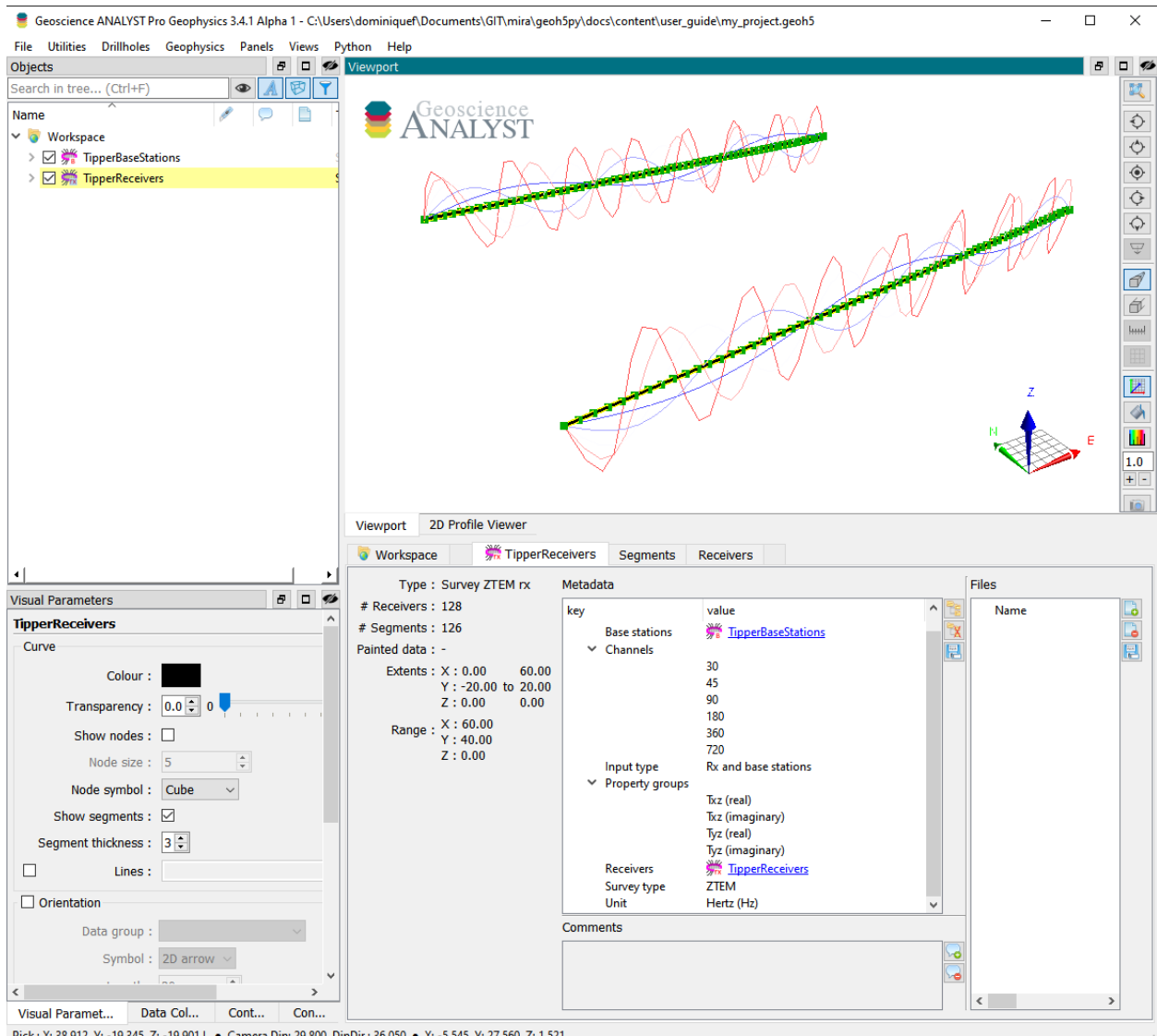
Unit

Units for frequency sampling of the data: Hertz (Hz), KiloHertz (kHz), MegaHertz (MHz) or Gigahertz (GHz).

Tipper

This object can be used to store tipper (ZTEM) surveys - a natural-source geophysical method. Data are provided in the frequency-domain as point source measurements of tipper data.

The following example shows how to generate a tipper survey with associated data stored in geoh5 format and accessible from [Geoscience ANALYST](#).



```
[1]: import numpy as np
from geoh5py.workspace import Workspace
from geoh5py.objects import TipperReceivers, TipperBaseStations

# Create a new project
workspace = Workspace("my_project.geoh5")

# Define the pole locations
n_stations = 64
n_lines = 2
x_loc, y_loc = np.meshgrid(np.linspace(0, 60, n_stations), np.linspace(-20, 20., n_lines))
vertices = np.c_[x_loc.ravel(), y_loc.ravel(), np.zeros_like(x_loc).ravel()]

# Assign a line ID to the poles (vertices)
parts = np.kron(np.arange(n_lines), np.ones(n_stations)).astype('int')
```

(continues on next page)

(continued from previous page)

```
# Create the survey from vertices
receivers = TipperReceivers.create(workspace, vertices=vertices, parts=parts)
base = TipperBaseStations.create(workspace, vertices=vertices)
```

We have so far created two separate entities, one for the receiver locations and another for the base station(s). In order to finalize the survey, the association must be made between the two entities:

```
[2]: receivers.base_station = base
```

or equivalently

```
[3]: base.receivers = receivers
```

Only one of the two options above is needed.

Metadata

Along with the *TipperReceivers*, the metadata contains all the necessary information to define the geophysical experiment.

```
[4]: receivers.metadata
```

```
[4]: {'EM Dataset': {'Base stations': UUID('a8a478ea-602c-4a52-96a9-739c1a70512e'),
  'Channels': [],
  'Input type': 'Rx and base stations',
  'Property groups': [],
  'Receivers': UUID('405fc56d-8dea-4109-b0c8-af46b95acb36'),
  'Survey type': 'ZTEM',
  'Unit': 'Hertz (Hz)'}]}
```

Channels

List of frequencies at which the data are provided.

```
[5]: receivers.channels = [30., 45., 90., 180., 360., 720.]
```

Input type

Generic label used in the geoh5 standard for all EM survey entities. Restricted to `Rx` and `base station` in the case of a tipper survey.

Property groups

List of *PropertyGroups* defining the various data components (e.g. Txz (real), Tyz (imag), ...). It is not required to supply all components of the impedance tensor, but it is expected that each component contains a list of data channels of length and in the same order as the Channels (one Data per frequency).

The class method *add_components_data* can help users add data from nested dictionaries. Below is an example using four components:

```
[6]: # Arbitrary data generator using sine functions
data_fun = lambda c, f: (c+1.) * (f+1.) * np.sin(f * np.pi * (x_loc * y_loc).ravel() / 400.)

# Create a nested dictionary of component and frequency data.
data = {
    component : {
        f"{component}_{freq}": {"values": data_fun(cc, ff)} for ff, freq in
    enumerate(receivers.channels)
    } for cc, component in enumerate([
        "Txz (real)", "Txz (imaginary)",
        "Tyz (real)", "Tyz (imaginary)",
    ])
}

receivers.add_components_data(data)
```

```
[6]: [<geoh5py.groups.property_group.PropertyGroup at 0x7f395d8afc90>,
<geoh5py.groups.property_group.PropertyGroup at 0x7f395d803910>,
<geoh5py.groups.property_group.PropertyGroup at 0x7f395d815f90>,
<geoh5py.groups.property_group.PropertyGroup at 0x7f395d819250>]
```

Metadata are updated immediately to reflect the addition of components:

```
[7]: receivers.metadata
```

```
[7]: {'EM Dataset': {'Base stations': UUID('a8a478ea-602c-4a52-96a9-739c1a70512e'),
'Channels': [30.0, 45.0, 90.0, 180.0, 360.0, 720.0],
'Input type': 'Rx and base stations',
'Property groups': ['Txz (real)',
'Txz (imaginary)',
'Tyz (real)',
'Tyz (imaginary)'],
'Receivers': UUID('405fc56d-8dea-4109-b0c8-af46b95acb36'),
'Survey type': 'ZTEM',
'Unit': 'Hertz (Hz)'}]}
```

Data channels associated with each component can be quickly accessed through the *BaseEMSurvey.components* property:

```
[8]: receivers.components
```

```
[8]: {'Txz (real)': [<geoh5py.data.float_data.FloatData at 0x7f395d80bf90>,
<geoh5py.data.float_data.FloatData at 0x7f395eb744d0>,
<geoh5py.data.float_data.FloatData at 0x7f395d879c10>,
<geoh5py.data.float_data.FloatData at 0x7f395d879a90>,
```

(continues on next page)

(continued from previous page)

```

<geoh5py.data.float_data.FloatData at 0x7f395d879cd0>,
<geoh5py.data.float_data.FloatData at 0x7f395d8af1d0>],
'Txz (imaginary)': [<geoh5py.data.float_data.FloatData at 0x7f397a7f5d10>,
<geoh5py.data.float_data.FloatData at 0x7f395d8af110>,
<geoh5py.data.float_data.FloatData at 0x7f395d803510>,
<geoh5py.data.float_data.FloatData at 0x7f395d803710>,
<geoh5py.data.float_data.FloatData at 0x7f395d803350>,
<geoh5py.data.float_data.FloatData at 0x7f395d803790>],
'Tyz (real)': [<geoh5py.data.float_data.FloatData at 0x7f395d803a10>,
<geoh5py.data.float_data.FloatData at 0x7f395d8158d0>,
<geoh5py.data.float_data.FloatData at 0x7f395d815910>,
<geoh5py.data.float_data.FloatData at 0x7f395d815a90>,
<geoh5py.data.float_data.FloatData at 0x7f395d815890>,
<geoh5py.data.float_data.FloatData at 0x7f395d815bd0>],
'Tyz (imaginary)': [<geoh5py.data.float_data.FloatData at 0x7f395d819f10>,
<geoh5py.data.float_data.FloatData at 0x7f395d819990>,
<geoh5py.data.float_data.FloatData at 0x7f395d819a50>,
<geoh5py.data.float_data.FloatData at 0x7f395d819c10>,
<geoh5py.data.float_data.FloatData at 0x7f395d819d90>,
<geoh5py.data.float_data.FloatData at 0x7f395d8197d0>]]}

```

Receivers

Generic label used in the geoh5 standard for EM survey to identify the *TipperReceivers* entity.

Base stations

Generic label used in the geoh5 standard for EM survey to identify the *TipperBaseStations* entity.

Survey type

Label identifier for ZTEM survey type.

Unit

Units for frequency sampling of the data: Hertz (Hz), KiloHertz (kHz), MegaHertz (MHz) or Gigahertz (GHz).

```
[9]: workspace.finalize()
```

```

/home/docs/checkouts/readthedocs.org/user_builds/geoh5py/conda/v0.6.0/lib/python3.7/site-
packages/geoh5py/workspace/workspace.py:854: UserWarning: The 'finalize' method will
be deprecated in future versions of geoh5py in favor of `workspace.close()`. Please
update your code to suppress this warning.
"The 'finalize' method will be deprecated in future versions of geoh5py in"

```

2.3 geoh5py

2.3.1 geoh5py.data

geoh5py.data.blob_data

```
class geoh5py.data.blob_data.BlobData(data_type: DataType, **kwargs)
    Bases: Data
    classmethod primitive_type() → PrimitiveTypeEnum
```

geoh5py.data.color_map

```
class geoh5py.data.color_map.ColorMap(**kwargs)
    Bases: object
    Records colors assigned to value ranges (where Value is the start of the range).
    property name: str
        str: Name of the colormap
    property parent
        Parent data type
    property values: np.ndarray | None
        numpy.array: Colormap defined by values and corresponding RGBA:
```

```
values = [
    [V_1, R_1, G_1, B_1, A_1],
    ..., [V_i, R_i, G_i, B_i, A_i]
]
```

where V (Values) are sorted floats defining the position of each RGBA. R (Red), G (Green), B (Blue) and A (Alpha) are integer values between [0, 255].

geoh5py.data.data

```
class geoh5py.data.data.Data(data_type: DataType, **kwargs)
    Bases: Entity
    Base class for Data entities.
    add_file(file: str)
        Alias not implemented from base Entity class.
    property association: DataAssociationEnum | None
        DataAssociationEnum: Relationship made between the values() and elements of the parent object.
        Association can be set from a str chosen from the list of available DataAssociationEnum options.
    property entity_type: DataType
        DataType
    property modifiable: bool
        bool Entity can be modified.
```

property n_values: `int | None`

`int`: Number of expected data values based on *association*

abstract classmethod primitive_type() \rightarrow *PrimitiveTypeEnum*

remove_data_from_group(*data*: `list | Entity | uuid.UUID | str`, *name*: `str | None = None`)

Remove self from a property group.

property values

Data values

geoh5py.data.data_association_enum

class geoh5py.data.data_association_enum.**DataAssociationEnum**(*value*)

Bases: `Enum`

Known data association between *values* and the *parent* object. Available options:

CELL = 2

DEPTH = 6

FACE = 4

GROUP = 5

OBJECT = 1

UNKNOWN = 0

VERTEX = 3

geoh5py.data.data_type

class geoh5py.data.data_type.**DataType**(*workspace*: `workspace.Workspace`, ***kwargs*)

Bases: *EntityType*

DataType class

property color_map: *ColorMap* | `None`

ColorMap: Colormap used for plotting

The colormap can be set from a dict of sorted values with corresponding RGBA color.

```
color_map = {
    val_1: [r_1, g_1, b_1, a_1],
    ...,
    val_i: [r_i, g_i, b_i, a_i]
}
```

classmethod create(*workspace*: `workspace.Workspace`, *data_class*: `type[data.Data]`) \rightarrow *DataType*

Creates a new instance of *DataType* with corresponding *PrimitiveTypeEnum*.

Parameters

data_class – A *Data* implementation class.

Returns

A new instance of *DataType*.

classmethod **find_or_create**(*workspace*: *workspace.Workspace*, ***kwargs*) → *DataType*

Find or creates an EntityType with given UUID that matches the given Group implementation class.

Parameters

workspace – An active Workspace class

Returns

A new instance of *DataType*.

classmethod **for_x_data**(*workspace*: *workspace.Workspace*) → *DataType*

classmethod **for_y_data**(*workspace*: *workspace.Workspace*) → *DataType*

classmethod **for_z_data**(*workspace*: *workspace.Workspace*) → *DataType*

property **hidden**: *bool*

bool: Hidden data [False]

property **mapping**: *str*

str: Color stretching type chosen from: 'linear', ['equal_area'], 'logarithmic', 'cdf', 'missing'

property **number_of_bins**: *int* | *None*

int: Number of bins used by the histogram [50]

property **primitive_type**: *PrimitiveTypeEnum* | *None*

PrimitiveTypeEnum

property **transparent_no_data**: *bool*

bool: Use transparent for no-data-value [True]

property **units**: *str* | *None*

str: Data units

property **value_map**: *ReferenceValueMap* | *None*

ReferenceValueMap: Reference value map for ReferenceData

The value_map can be set from a dict of sorted values with corresponding str description.

```
value_map = {
    val_1: str_1,
    ...,
    val_i: str_i
}
```

geoh5py.data.data_unit

class **geoh5py.data.data_unit.DataUnit**(*unit_name*: *str* | *None* = *None*)

Bases: *object*

Data unit

property **name**: *str* | *None*

geoh5py.data.datetime_data

class geoh5py.data.datetime_data.**DatetimeData**(data_type: *DataType*, **kwargs)

Bases: *Data*

classmethod **primitive_type**() → *PrimitiveTypeEnum*

geoh5py.data.filename_data

class geoh5py.data.filename_data.**FilenameData**(data_type: *DataType*, file_name=None, **kwargs)

Bases: *Data*

property **file_name**: str | None

str Text value.

classmethod **primitive_type**() → *PrimitiveTypeEnum*

save_file(path: str = './', name=None)

Save the file to disk.

Parameters

- **path** – Directory to save the file to.
- **name** – Name given to the file.

property values: bytes | None

Binary str value representation of a file.

geoh5py.data.float_data

class geoh5py.data.float_data.**FloatData**(data_type: *DataType*, **kwargs)

Bases: *NumericData*

Data container for floats values

property **ndv**: float

No-Data-Value

classmethod **primitive_type**() → *PrimitiveTypeEnum*

geoh5py.data.geometric_data_constants

class geoh5py.data.geometric_data_constants.**GeometricDataConstants**

Bases: object

classmethod **primitive_type**() → *PrimitiveTypeEnum*

classmethod **x_datatype_uid**() → UUID

classmethod **y_datatype_uid**() → UUID

classmethod **z_datatype_uid**() → UUID

geoh5py.data.integer_data

class geoh5py.data.integer_data.IntegerData(*data_type: DataType, **kwargs*)

Bases: *NumericData*

property ndv: int

No-Data-Value

classmethod primitive_type() → *PrimitiveTypeEnum*

property values: np.ndarray | None

Returns

values: An array of integer values

geoh5py.data.numeric_data

class geoh5py.data.numeric_data.NumericData(*data_type: DataType, **kwargs*)

Bases: *Data*, ABC

Data container for floats values

check_vector_length(*values*) → ndarray

Check for possible mismatch between the length of values stored and the expected number of cells or vertices.

abstract property ndv

No-data-value

classmethod primitive_type() → *PrimitiveTypeEnum*

property values: np.ndarray | None

Returns

values: An array of float values

geoh5py.data.primitive_type_enum

class geoh5py.data.primitive_type_enum.PrimitiveTypeEnum(*value*)

Bases: Enum

Known data type.

Available options:

BLOB = 6

DATETIME = 8

FILENAME = 5

FLOAT = 2

GEOMETRIC = 9

INTEGER = 1

```

INVALID = 0

MULTI_TEXT = 10

REFERENCED = 4

TEXT = 3

VECTOR = 7

```

geoh5py.data.reference_value_map

class geoh5py.data.reference_value_map.**ReferenceValueMap**(*color_map: dict[int, str] | None = None*)

Bases: `ABC`

Maps from reference index to reference value of ReferencedData.

property map

dict: A reference dictionary mapping values to strings

geoh5py.data.referenced_data

class geoh5py.data.referenced_data.**ReferencedData**(*data_type: DataType, **kwargs*)

Bases: `IntegerData`

Reference data described by indices and associated strings.

classmethod primitive_type() → `PrimitiveTypeEnum`

property value_map

Pointer to the data.data_type.DataType.value_map

geoh5py.data.text_data

class geoh5py.data.text_data.**CommentsData**(*data_type: DataType, **kwargs*)

Bases: `Data`

Comments added to an Object or Group. Stored as a list of dictionaries with the following keys:

```

comments = [
    {
        "Author": "username",
        "Date": "2020-05-21T10:12:15",
        "Text": "A text comment."
    },
]

```

classmethod primitive_type() → `PrimitiveTypeEnum`

property values: list[dict] | None

list List of comments

class geoh5py.data.text_data.**MultiTextData**(*data_type: DataType, **kwargs*)

Bases: `Data`

classmethod `primitive_type()` → *PrimitiveTypeEnum*

property values: `np.ndarray | str | None`

`str` Text value.

class `geoh5py.data.text_data.TextData(data_type: DataType, **kwargs)`

Bases: *Data*

classmethod `primitive_type()` → *PrimitiveTypeEnum*

property values: `np.ndarray | str | None`

`str` Text value.

`geoh5py.data.unknown_data`

class `geoh5py.data.unknown_data.UnknownData(data_type: DataType, association: DataAssociationEnum, name: str, uid: uuid.UUID | None = None)`

Bases: *Data*

classmethod `primitive_type()` → *PrimitiveTypeEnum*

2.3.2 `geoh5py.groups`

`geoh5py.groups.container_group`

class `geoh5py.groups.container_group.ContainerGroup(group_type: GroupType, **kwargs)`

Bases: *Group*

The type for the basic Container group.

classmethod `default_type_uid()` → *UUID*

`geoh5py.groups.custom_group`

class `geoh5py.groups.custom_group.CustomGroup(group_type: GroupType, **kwargs)`

Bases: *Group*

A custom group, for an unlisted Group type.

classmethod `default_type_uid()` → *uuid.UUID* | *None*

`geoh5py.groups.drillhole_group`

class `geoh5py.groups.drillhole_group.DrillholeGroup(group_type: GroupType, name='Drillholes Group', **kwargs)`

Bases: *Group*

The type for the group containing drillholes.

classmethod `default_type_uid()` → *UUID*


```
class geoh5py.groups.drillhole_group.IntegratorDrillholeGroup(group_type: GroupType,
                                                             **kwargs)
```

Bases: [DrillholeGroup](#)

The type for the group containing drillholes.

classmethod `default_type_uid()` → UUID

geoh5py.groups.giftools_group

```
class geoh5py.groups.giftools_group.GiftoolsGroup(group_type: GroupType, **kwargs)
```

Bases: [Group](#)

The type for a GIFtools group.

classmethod `default_type_uid()` → UUID

geoh5py.groups.group

```
class geoh5py.groups.group.Group(group_type: GroupType, **kwargs)
```

Bases: [Entity](#)

Base Group class

add_comment(comment: str, author: str | None = None)

Add text comment to an object.

Parameters

- **comment** – Text to be added as comment.
- **author** – Author’s name or contributors.

property comments

Fetch a [CommentsData](#) entity from children.

copy_from_extent(bounds: np.ndarray, parent=None, copy_children: bool = True) → [Group](#) | None

Find indices of vertices within a rectangular bounds.

Parameters

- **bounds** – shape(2, 2) Bounding box defined by the South-West and North-East coordinates. Extents can also be provided as 3D coordinates with shape(2, 3) defining the top and bottom limits.
- **attributes** – Dictionary of attributes to clip by extent.

abstract classmethod `default_type_uid()` → uuid.UUID | None

property `entity_type:` [GroupType](#)

property extent

Bounding box 3D coordinates defining the limits of the entity.

classmethod `find_or_create_type(workspace: workspace.Workspace, **kwargs)` → [GroupType](#)

geoh5py.groups.group_type

```
class geoh5py.groups.group_type.GroupType(workspace: workspace.Workspace, **kwargs)
    Bases: EntityType

    property allow_delete_content: bool
        bool: [True] Allow to delete the group children.

    property allow_move_content: bool
        bool: [True] Allow to move the group children.

    static create_custom(workspace: workspace.Workspace, **kwargs) → GroupType
        Creates a new instance of GroupType for an unlisted custom Group type with a new auto-generated UUID.

    classmethod find_or_create(workspace: workspace.Workspace, entity_class, **kwargs) → GroupType
        Find or creates an EntityType with given UUID that matches the given Group implementation class.

    Parameters
        • workspace – An active Workspace class
        • entity_class – An Group implementation class.

    Returns
        A new instance of GroupType.
```

geoh5py.groups.integrator_group

```
class geoh5py.groups.integrator_group.AirborneTheme(group_type: GroupType, **kwargs)
    Bases: Group

    The type for a INTEGRATOR Airborne Theme.

    classmethod default_type_uid() → UUID

class geoh5py.groups.integrator_group.EarthModelsTheme(group_type: GroupType, **kwargs)
    Bases: Group

    The type for a INTEGRATOR Earth Models Theme.

    classmethod default_type_uid() → UUID

class geoh5py.groups.integrator_group.GeochemistryMineralogyDataSet(group_type: GroupType,
                                                                       **kwargs)
    Bases: Group

    The type for a INTEGRATOR Geochemistry & Mineralogy DataSet.

    classmethod default_type_uid() → UUID

class geoh5py.groups.integrator_group.GeochemistryMineralogyTheme(group_type: GroupType,
                                                                      **kwargs)
    Bases: Group

    The type for a INTEGRATOR Geochemistry & Mineralogy Theme.

    classmethod default_type_uid() → UUID
```

```
class geoh5py.groups.integrator_group.GeophysicsTheme(group_type: GroupType, **kwargs)
```

Bases: [Group](#)

The type for a INTEGRATOR Geophysics Theme.

```
classmethod default_type_uid() → UUID
```

```
class geoh5py.groups.integrator_group.GroundTheme(group_type: GroupType, **kwargs)
```

Bases: [Group](#)

The type for a INTEGRATOR Ground Theme.

```
classmethod default_type_uid() → UUID
```

```
class geoh5py.groups.integrator_group.IntegratorGroup(group_type: GroupType, **kwargs)
```

Bases: [Group](#)

The type for a INTEGRATOR group.

```
classmethod default_type_uid() → UUID
```

```
class geoh5py.groups.integrator_group.IntegratorProject(group_type: GroupType, **kwargs)
```

Bases: [Group](#)

The type for a INTEGRATOR group.

```
classmethod default_type_uid() → UUID
```

```
class geoh5py.groups.integrator_group.ObservationPointsTheme(group_type: GroupType, **kwargs)
```

Bases: [Group](#)

The type for a INTEGRATOR Observation Points Theme.

```
classmethod default_type_uid() → UUID
```

```
class geoh5py.groups.integrator_group.QueryGroup(group_type: GroupType, **kwargs)
```

Bases: [Group](#)

The type for a INTEGRATOR Query Group.

```
classmethod default_type_uid() → UUID
```

```
class geoh5py.groups.integrator_group.RockPropertiesTheme(group_type: GroupType, **kwargs)
```

Bases: [Group](#)

The type for a INTEGRATOR Rock Properties Theme.

```
classmethod default_type_uid() → UUID
```

```
class geoh5py.groups.integrator_group.SamplesTheme(group_type: GroupType, **kwargs)
```

Bases: [Group](#)

The type for a INTEGRATOR Samples Theme.

```
classmethod default_type_uid() → UUID
```

geoh5py.groups.maps_group

```
class geoh5py.groups.maps_group.MapsGroup(group_type: GroupType, **kwargs)
```

Bases: *Group*

The type for the basic Container group.

classmethod `default_type_uid()` → UUID

geoh5py.groups.notype_group

```
class geoh5py.groups.notype_group.NoTypeGroup(group_type: GroupType, **kwargs)
```

Bases: *Group*

A group with no type.

classmethod `default_type_uid()` → UUID

geoh5py.groups.property_group

```
class geoh5py.groups.property_group.PropertyGroup(parent: ObjectBase, **kwargs)
```

Bases: ABC

Property group listing data children of an object. This group is not registered to the workspace and only visible to the parent object.

property `association`: *DataAssociationEnum*

DataAssociationEnum Data association

property `attribute_map`: dict

dict Attribute names mapping between geoh5 and geoh5py

property `name`: str

str Name of the group

property `parent`: *Entity*

The parent *ObjectBase*

property `properties`: list[uuid.UUID]

List of unique identifiers for the *Data* contained in the property group.

property `property_group_type`: str

property `uid`: UUID

uuid.UUID Unique identifier

geoh5py.groups.root_group

class geoh5py.groups.root_group.**RootGroup**(group_type: GroupType, **kwargs)

Bases: *NoTypeGroup*

The Root group of a workspace.

property parent

Parental entity of root is always None

geoh5py.groups.simpeg_group

class geoh5py.groups.simpeg_group.**SimPEGGroup**(group_type: GroupType, **kwargs)

Bases: *Group*

Group for SimPEG inversions.

classmethod default_type_uid() → UUID

property options: dict | None

Metadata attached to the entity.

geoh5py.groups.survey_group

class geoh5py.groups.survey_group.**AirborneGeophysics**(group_type: GroupType, **kwargs)

Bases: *Group*

The type for the basic Container group.

classmethod default_type_uid() → UUID

2.3.3 geoh5py.io

geoh5py.io.h5_reader

class geoh5py.io.h5_reader.**H5Reader**

Bases: object

Class to read information from a geoh5 file.

classmethod fetch_array_attribute(file: str | h5py.File, uid: uuid.UUID, entity_type: str, key: str) → np.ndarray | None

Get an entity attribute stores as array such as *cells*.

Parameters

- **file** – h5py.File or name of the target geoh5 file
- **uid** – Unique identifier of the target object.
- **entity_type** – Group type to fetch entity from.
- **key** – Field attribute name.

Return cells

numpy.ndarray of int.

classmethod `fetch_attributes(file: str | h5py.File, uid: uuid.UUID, entity_type: str) → tuple[dict, dict, dict] | None`

Get attributes of an [Entity](#).

Parameters

- **file** – `h5py.File` or name of the target geoh5 file
- **uid** – Unique identifier
- **entity_type** – Type of entity from ‘group’, ‘data’, ‘object’, ‘group_type’, ‘data_type’, ‘object_type’

Returns

attributes: dict of attributes for the [Entity](#)
type_attributes: dict of attributes for the `EntityType`
property_groups: dict of data `uuid.UUID`

classmethod `fetch_children(file: str | h5py.File, uid: uuid.UUID, entity_type: str) → dict`

Get [children](#) of an [Entity](#).

Parameters

- **file** – `h5py.File` or name of the target geoh5 file
- **uid** – Unique identifier
- **entity_type** – Type of entity from ‘group’, ‘data’, ‘object’, ‘group_type’, ‘data_type’, ‘object_type’

Return children

[{uid: type}, ...] List of dictionaries for the children uid and type

classmethod `fetch_concatenated_attributes(file: str | h5py.File, uid: uuid.UUID, entity_type: str, label: str) → list | dict | None`

Get ‘Attributes’, ‘Data’ or ‘Index’ from Concatenator group.

Parameters

- **file** – `h5py.File` or name of the target geoh5 file
- **uid** – Unique identifier
- **entity_type** – Type of entity from ‘group’, ‘data’, ‘object’, ‘group_type’, ‘data_type’, ‘object_type’
- **label** – Group identifier for the attribute requested.

Return children

[{uid: type}, ...] List of dictionaries for the children uid and type

classmethod `fetch_concatenated_values(file: str | h5py.File, uid: uuid.UUID, entity_type: str, label: str) → tuple | None`

Get [children](#) values of concatenated group.

Parameters

- **file** – `h5py.File` or name of the target geoh5 file
- **uid** – Unique identifier
- **entity_type** – Type of entity from ‘group’, ‘data’, ‘object’, ‘group_type’, ‘data_type’, ‘object_type’

- **label** – Group identifier for the attribute requested.

Return children

[{uuid: type}, ...] List of dictionaries for the children uuid and type

classmethod **fetch_file_object**(*file: str | h5py.File, uid: uuid.UUID, file_name: str*) → bytes | None

Load data associated with an image file

Parameters

- **file** – Name of the target geoh5 file
- **uid** – Unique identifier of the target entity
- **file_name** – Name of the file stored as bytes data.

Return values

Data file stored as bytes

classmethod **fetch_metadata**(*file: str | h5py.File, uid: uuid.UUID, entity_type: str = 'Objects', argument: str = 'Metadata'*) → str | dict | None

Fetch text of dictionary type attributes of an entity.

Parameters

- **file** – Target h5 file.
- **uid** – Unique identifier of the target Entity.
- **entity_type** – Base type of the target Entity.
- **argument** – Label name of the dictionary.

classmethod **fetch_project_attributes**(*file: str | h5py.File*) → dict[Any, Any]

Get attributes of an *Entity*.

Parameters

file – h5py.File or name of the target geoh5 file

Return attributes

dict of attributes.

classmethod **fetch_property_groups**(*file: str | h5py.File, uid: uuid.UUID*) → dict[str, dict[str, str]]

Get the property groups.

Parameters

- **file** – h5py.File or name of the target geoh5 file
- **uid** – Unique identifier of the target entity

Return property_group_attributes

dict of property groups and respective attributes.

```
property_group = {
    "group_1": {"attribute": value, ...},
    ...,
    "group_N": {"attribute": value, ...},
}
```

classmethod **fetch_type**(*file: str | h5py.File, uid: uuid.UUID, entity_type: str*) → dict

Fetch a type from the target geoh5.

Parameters

- **file** – `h5py.File` or name of the target geoh5 file
- **uid** – Unique identifier of the target entity
- **entity_type** – One of ‘Data’, ‘Object’ or ‘Group’

Return property_group_attributes

dict of property groups and respective attributes.

classmethod **fetch_type_attributes**(*type_handle: Group*) → dict

Fetch type attributes from a given h5 handle.

classmethod **fetch_uuids**(*file: str | h5py.File, entity_type: str*) → list

Fetch all uuids of a given type from geoh5

Parameters

- **file** – `h5py.File` or name of the target geoh5 file
- **entity_type** – Type of entity from ‘group’, ‘data’, ‘object’, ‘group_type’, ‘data_type’, ‘object_type’

Return uuids

[uuid1, uuid2, ...] List of uuids

classmethod **fetch_value_map**(*h5_handle: Group*) → dict

Get data value_map

Parameters

h5_handle – Handle to the target h5 group.

Return value_map

dict of {int: str}

classmethod **fetch_values**(*file: str | h5py.File, uid: uuid.UUID*) → np.ndarray | str | float | None

Get data [values](#)

Parameters

- **file** – `h5py.File` or name of the target geoh5 file
- **uid** – Unique identifier of the target entity

Return values

`numpy.array` of float

static **format_type_string**(*string: str*) → str

Format names used for types.

geoh5py.io.h5_writer

class `geoh5py.io.h5_writer.H5Writer`

Bases: `object`

Writing class to a geoh5 file.

classmethod **clear_stats_cache**(*file: str | h5py.File, entity: Data*) → None

Clear the StatsCache dataset.

Parameters

- **file** – Name or handle to a geoh5 file.

- **entity** – Target entity.

classmethod `create_dataset(entity_handle, dataset: ndarray, label: str) → None`

Create a dataset on geoh5.

Parameters

- **entity_handle** – Pointer to a hdf5 group
- **dataset** – Array of values to be written
- **label** – Name of the dataset on file

classmethod `create_geoh5(file: str | h5py.File, workspace: workspace.Workspace)`

Add the geoh5 core structure.

Parameters

- **file** – Name or handle to a geoh5 file.
- **workspace** – [Workspace](#) object defining the project structure.

Return h5file

Pointer to a geoh5 file.

classmethod `fetch_handle(file: str | h5py.File, entity, return_parent: bool = False) → None | h5py.Group`

Get a pointer to an [Entity](#) in geoh5.

Parameters

- **file** – Name or handle to a geoh5 file
- **entity** – Target [Entity](#)
- **return_parent** – Option to return the handle to the parent entity.

Return entity_handle

HDF5 pointer to an existing entity, parent or None if not found.

static `remove_child(file: str | h5py.File, uid: uuid.UUID, ref_type: str, parent: Entity) → None`

Remove a child from a parent.

Parameters

- **file** – Name or handle to a geoh5 file
- **uid** – uuid of the target [Entity](#)
- **ref_type** – Input type from: ‘Types’, ‘Groups’, ‘Objects’ or ‘Data’
- **parent** – Remove entity from parent.

static `remove_entity(file: str | h5py.File, uid: uuid.UUID, ref_type: str, parent: Entity | None = None) → None`

Remove an entity and its type from the target geoh5 file.

Parameters

- **file** – Name or handle to a geoh5 file
- **uid** – uuid of the target [Entity](#)
- **ref_type** – Input type from: ‘Types’, ‘Groups’, ‘Objects’ or ‘Data’
- **parent** – Remove entity from parent.

classmethod `save_entity(file: str | h5py.File, entity, add_children: bool = True) → h5py.Group`

Write an *Entity* to geoh5 with its *children*.

Parameters

- **file** – Name or handle to a geoh5 file.
- **entity** – Target *Entity*.
- **add_children** – Add *children*.

`str_type = dtype('O')`

classmethod `update_concatenated_field(file: str | h5py.File, entity, attribute: str, channel: str) → None`

Update the attributes of a concatenated *Entity*.

Parameters

- **file** – Name or handle to a geoh5 file.
- **entity** – Target *Entity*.
- **attribute** – Name of the attribute to get updated.
- **channel** – Name of the data or index to be modified.

classmethod `update_field(file: str | h5py.File, entity, attribute: str, **kwargs) → None`

Update the attributes of an *Entity*.

Parameters

- **file** – Name or handle to a geoh5 file.
- **entity** – Target *Entity*.
- **attribute** – Name of the attribute to get updated.

classmethod `write_array_attribute(file: str | h5py.File, entity, attribute, values=None, **kwargs) → None`

Add surveys of an object.

Parameters

- **file** – Name or handle to a geoh5 file.
- **entity** – Target entity.
- **attribute** – Name of the attribute to be written to geoh5

classmethod `write_attributes(file: str | h5py.File, entity) → None`

Write attributes of an *Entity*.

Parameters

- **file** – Name or handle to a geoh5 file.
- **entity** – Entity with attributes to be added to the geoh5 file.

classmethod `write_color_map(file: str | h5py.File, entity_type: shared.EntityType) → None`

Add *ColorMap* to a *DataType*.

Parameters

- **file** – Name or handle to a geoh5 file
- **entity_type** – Target entity_type with color_map

classmethod `write_data_values(file: str | h5py.File, entity, attribute, values=None, **kwargs) → None`

Add data `values`.

Parameters

- **file** – Name or handle to a geoh5 file.
- **entity** – Target entity.
- **attribute** – Name of the attribute to be written to geoh5

classmethod `write_entity(file: str | h5py.File, entity) → h5py.Group`

Add an `Entity` and its attributes to geoh5. The function returns a pointer to the entity if already present on file.

Parameters

- **file** – Name or handle to a geoh5 file.
- **entity** – Target `Entity`.

Return entity

Pointer to the written entity. Active link if “close_file” is False.

classmethod `write_entity_type(file: str | h5py.File, entity_type: shared.EntityType) → h5py.Group`

Add an `EntityType` to geoh5.

Parameters

- **file** – Name or handle to a geoh5 file.
- **entity_type** – Entity with type to be added.

Return type

Pointer to `EntityType` in geoh5.

classmethod `write_file_name_data(entity_handle: Group, entity: FilenameData, values: bytes) → None`

Write a dataset for the file name and file blob.

Parameters

- **entity_handle** – Pointer to the geoh5 Group.
- **entity** – Target `FilenameData` entity.
- **values** – Bytes data

classmethod `write_properties(file: str | h5py.File, entity: Entity) → None`

Add properties of an `Entity`.

Parameters

- **file** – Name or handle to a geoh5 file.
- **entity** – Target `Entity`.

classmethod `write_property_groups(file: str | h5py.File, entity) → None`

Write `PropertyGroup` associated with an `Entity`.

Parameters

- **file** – Name or handle to a geoh5 file.
- **entity** – Target `Entity`.

classmethod `write_to_parent(file: str | h5py.File, entity: Entity, recursively=False) → None`

Add/create an *Entity* and add it to its parent.

Parameters

- **file** – Name or handle to a geoh5 file.
- **entity** – Entity to be added or linked to a parent in geoh5.
- **recursively** – Add parents recursively until reaching the *RootGroup*.

classmethod `write_value_map(file: str | h5py.File, entity_type: shared.EntityType) → None`

Add *ReferenceValueMap* to a *Data* type.

Parameters

- **file** – Name or handle to a geoh5 file
- **entity_type** – Target entity_type with value_map

classmethod `write_visible(file: str | h5py.File, entity) → None`

Needs revision once Visualization is implemented

Parameters

- **file** – Name or handle to a geoh5 file
- **entity** – Target entity

2.3.4 geoh5py.objects

`geoh5py.objects.surveys`

`geoh5py.objects.surveys.electromagnetics`

`geoh5py.objects.surveys.electromagnetics.airborne_tem`

class `geoh5py.objects.surveys.electromagnetics.airborne_tem.AirborneTEMReceivers(object_type: Object-
Type,
name='Airborne
TEM Rx',
**kwargs)`

Bases: *BaseAirborneTEM*

Airborne time-domain electromagnetic receivers class.

classmethod `default_type_uid() → UUID`

Returns

Default unique identifier

property `type`

Survey element type

```
class geoh5py.objects.surveys.electromagnetics.airborne_tem.AirborneTEMTransmitters(object_type:
    Ob-
    ject-
    Type,
    name='Airborne
    TEM
    Tx',
    **kwargs)
```

Bases: [BaseAirborneTEM](#)

Airborne time-domain electromagnetic transmitters class.

classmethod `default_type_uid()` → UUID

Returns

Default unique identifier

property `type`

Survey element type

```
class geoh5py.objects.surveys.electromagnetics.airborne_tem.BaseAirborneTEM(object_type:
    Object
    Type,
    name='Curve',
    **kwargs)
```

Bases: [BaseTEMSurvey](#), [Curve](#)

property `crossline_offset: float | uuid.UUID | None`

Numeric value or property UUID for the crossline offset between receiver and transmitter.

property `default_input_types: list[str]`

Choice of survey creation types.

property `default_metadata: dict`

Default dictionary of metadata for AirborneTEM entities.

property `default_receiver_type`

Returns

Transmitter class

property `default_transmitter_type`

Returns

Transmitter class

fetch_metadata(*key: str*) → float | uuid.UUID | None

Fetch entry from the metadata.

property `inline_offset: float | uuid.UUID | None`

Numeric value or property UUID for the inline offset between receiver and transmitter.

property `loop_radius: float | None`

Transmitter loop radius

property `pitch: float | uuid.UUID | None`

Numeric value or property UUID for the pitch angle of the transmitter loop.

property `relative_to_bearing: bool | None`

Data relative_to_bearing

property roll: float | uuid.UUID | None

Numeric value or property UUID for the roll angle of the transmitter loop.

set_metadata(key: str, value: float | uuid.UUID | None)

property vertical_offset: float | uuid.UUID | None

Numeric value or property UUID for the vertical offset between receiver and transmitter.

property yaw: float | uuid.UUID | None

Numeric value or property UUID for the yaw angle of the transmitter loop.

geoh5py.objects.surveys.electromagnetics.base

class geoh5py.objects.surveys.electromagnetics.base.**BaseEMSurvey**(*object_type: ObjectType*,
***kwargs*)

Bases: *ObjectBase*, ABC

A base electromagnetics survey object.

add_components_data(*data: dict*) → list[*PropertyGroup*]

Add lists of data components to an EM survey. The name of each component is appended to the metadata 'Property groups'.

Data channels must be provided for every frequency or time in order specified by channels. The data channels can be supplied as either a list of *geoh5py.data.float_data.FloatData* entities or uuid.UUID

```
data = {
    "Component A": [
        data_entity_1,
        data_entity_2,
    ],
    "Component B": [...],
}
```

or a nested dictionary of arguments defining new Data entities as defined by the *add_data()* method.

```
data = {
    "Component A": {
        time_1: {
            'values': [v_11, v_12, ...],
            "entity_type": entity_type_A,
            ...,
        },
        time_2: {...},
        ...,
    },
    "Component B": {...},
}
```

Parameters

data – Dictionary of data components to be added to the survey.

Returns

List of property groups for all components added.

add_validate_component_data(*name: str, data_block: list | dict*)

Append a property group to the entity and its metadata after validations.

property channels

List of measured channels.

property components: dict | None

Rapid access to the list of data entities for all components.

copy(*parent=None, copy_children: bool = True, **kwargs*) → *BaseEMSurvey*

Function to copy a AirborneTEMReceivers to a different parent entity.

Parameters

- **parent** – Target parent to copy the entity under. Copied to current *parent* if None.
- **copy_children** – Create copies of AirborneTEMReceivers along with it.

Return entity

Registered AirborneTEMReceivers to the workspace.

abstract property default_input_types: list[str] | None

Input types. Must be one of 'Rx', 'Tx', 'Tx and Rx'.

property default_metadata

Default metadata structure. Implemented on the child class.

abstract property default_receiver_type: type

Returns

Receivers implemented on the child class.

abstract property default_transmitter_type: type

Returns

Transmitters implemented on the child class.

abstract classmethod default_type_uid() → UUID

Default unique identifier. Implemented on the child class.

abstract property default_units: list[str]

List of accepted units.

edit_metadata(*entries: dict[str, Any]*)

Utility function to edit or add metadata fields and trigger an update on the receiver and transmitter entities.

Parameters

entries – Metadata key value pairs.

property input_type: str | None

Data input type. Must be one of 'Rx', 'Tx' or 'Tx and Rx'

property metadata

Metadata attached to the entity.

property receivers: BaseEMSurvey | None

The associated TEM receivers.

property survey_type: str | None

Data input type. Must be one of 'Rx', 'Tx' or 'Tx and Rx'

property transmitters

The associated TEM transmitters (sources).

abstract property type

Survey element type

property unit: float | None

Default channel units for time or frequency defined on the child class.

```
class geoh5py.objects.surveys.electromagnetics.base.BaseTEMSurvey(object_type: ObjectType,
                                                                    **kwargs)
```

Bases: [BaseEMSurvey](#), ABC

abstract property default_input_types: list[str]

Input types for the survey element.

property default_units: list[str]

Accepted time units. Must be one of “Seconds (s)”, “Milliseconds (ms)”, “Microseconds (us)” or “Nanoseconds (ns)”

property timing_mark: float | None

Timing mark from the beginning of the discrete [waveform](#). Generally used as the reference (time=0.0) for the provided (-) on-time an (+) off-time channels.

property waveform: np.ndarray | None

Discrete waveform of the TEM source provided as `numpy.array` of type `float`, shape(n, 2)

```
waveform = [
    [time_1, current_1],
    [time_2, current_2],
    ...
]
```

property waveform_parameters: dict | None

Access the waveform parameters stored as a dictionary.

geoh5py.objects.surveys.electromagnetics.magnetotellurics

```
class geoh5py.objects.surveys.electromagnetics.magnetotellurics.MTReivers(object_type: ObjectType,
                                                                    name='Magnetotellurics
                                                                    rx', **kwargs)
```

Bases: [BaseEMSurvey](#), [Points](#)

A magnetotellurics survey object.

property default_input_types: list[str]

Choice of survey creation types.

property default_metadata: dict
Returns

Default unique identifier

property default_receiver_type

Returns

Transmitter class

property default_transmitter_type

Returns

Transmitter class

classmethod default_type_uid() → UUID

Returns

Default unique identifier

property default_units: list[str]

Accepted time units. Must be one of “Seconds (s)”, “Milliseconds (ms)”, “Microseconds (us)” or “Nanoseconds (ns)”

property type

Survey element type

geoh5py.objects.surveys.electromagnetics.tipper

class geoh5py.objects.surveys.electromagnetics.tipper.**BaseTipper**(*object_type*: [ObjectType](#),
base_stations:
[TipperBaseStations](#) | *None* =
None, ***kwargs*)

Bases: [BaseEMSurvey](#)

Base tipper survey class.

property base_stations: [TipperBaseStations](#) | None

The base station entity

property default_input_types: list[str]

Choice of survey creation types.

property default_metadata: dict

Returns

Default unique identifier

property default_receiver_type

Returns

Transmitter class

property default_transmitter_type

Returns

Transmitter class

property default_units: list[str]

Accepted time units. Must be one of “Seconds (s)”, “Milliseconds (ms)”, “Microseconds (us)” or “Nanoseconds (ns)”

```
class geoh5py.objects.surveys.electromagnetics.tipper.TipperBaseStations(object_type:
                                                                    ObjectType,
                                                                    name='Tipper base',
                                                                    **kwargs)
```

Bases: [BaseTipper](#), [Points](#)

A z-tipper EM survey object.

classmethod `default_type_uid()` → UUID

Returns

Default unique identifier

property type

Survey element type

```
class geoh5py.objects.surveys.electromagnetics.tipper.TipperReceivers(object_type: ObjectType,
                                                                    name='Tipper rx',
                                                                    **kwargs)
```

Bases: [BaseTipper](#), [Curve](#)

A z-tipper EM survey object.

classmethod `default_type_uid()` → UUID

Returns

Default unique identifier

property type

Survey element type

geoh5py.objects.surveys.direct_current

```
class geoh5py.objects.surveys.direct_current.CurrentElectrode(object_type: ObjectType,
                                                                **kwargs)
```

Bases: [PotentialElectrode](#)

Ground direct current electrode (transmitter).

add_default_ab_cell_id()

Utility function to set ab_cell_id's based on curve cells.

copy(parent=None, copy_children: bool = True, **kwargs)

Function to copy a survey to a different parent entity.

Parameters

- **parent** – Target parent to copy the entity under. Copied to current [parent](#) if None.
- **copy_children** – Create copies of all children entities along with it.

Return entity

Registered Entity to the workspace.

property current_electrodes

The associated current electrode object (sources).

classmethod `default_type_uid()` → UUID

Returns

Default unique identifier

property `potential_electrodes`: [PotentialElectrode](#) | None

The associated potential_electrodes (receivers)

class `geoh5py.objects.surveys.direct_current.PotentialElectrode`(*object_type*: [ObjectType](#),
***kwargs*)

Bases: [Curve](#)

Ground potential electrode (receiver).

property `ab_cell_id`: [ReferencedData](#) | None

Reference data entity mapping cells to a unique current dipole.

property `ab_map`: dict | None

Get the [ReferencedData.value_map](#) of the `ab_value_id`

copy(*parent*=None, *copy_children*: bool = True, ***kwargs*)

Function to copy a survey to a different parent entity.

Parameters

- **parent** – Target parent to copy the entity under. Copied to current [parent](#) if None.
- **copy_children** – Create copies of all children entities along with it.

Return entity

Registered Entity to the workspace.

property `current_electrodes`

The associated current electrode object (sources).

classmethod `default_type_uid()` → UUID

Returns

Default unique identifier

property `metadata`

Metadata attached to the entity.

property `potential_electrodes`

The associated potential_electrodes (receivers)

[geoh5py.objects.surveys.magnetics](#)

class `geoh5py.objects.surveys.magnetics.AirborneMagnetics`(*object_type*: [ObjectType](#), *name*='Curve',
***kwargs*)

Bases: [Curve](#)

An airborne magnetic survey object.

Warning: Partially implemented.

classmethod `default_type_uid()` → UUID

Returns

Default unique identifier

geoh5py.objects.block_model

class `geoh5py.objects.block_model.BlockModel`(*object_type*: `ObjectType`, ***kwargs*)

Bases: `ObjectBase`

Rectilinear 3D tensor mesh defined by three perpendicular axes. Each axis is divided into discrete intervals that define the cell dimensions. Nodal coordinates are determined relative to the origin and the sign of cell delimiters. Negative and positive cell delimiters are accepted to denote relative offsets from the origin.

property `cell_delimiters`

property `centroids`: `np.ndarray` | `None`

`numpy.array`, shape (`n_cells`, 3): Cell center locations in world coordinates.

```
centroids = [  
    [x_1, y_1, z_1],  
    ...,  
    [x_N, y_N, z_N]  
]
```

classmethod `default_type_uid()` → UUID

Returns

Default unique identifier

property `n_cells`: `int` | `None`

`int`: Total number of cells

property `origin`: `ndarray`

`numpy.array` of `float`, shape (3,): Coordinates of the origin.

property `rotation`: `float`

`float`: Clockwise rotation angle (degree) about the vertical axis.

property `shape`: `tuple` | `None`

list of `int`, len (3,): Number of cells along the u, v and z-axis

property `u_cell_delimiters`: `np.ndarray` | `None`

`numpy.array` of `float`: Nodal offsets along the u-axis relative to the origin.

property `u_cells`: `np.ndarray` | `None`

`numpy.array` of `float`, shape (`shape` [0],): Cell size along the u-axis.

property `v_cell_delimiters`: `np.ndarray` | `None`

`numpy.array` of `float`: Nodal offsets along the v-axis relative to the origin.

property `v_cells`: `np.ndarray` | `None`

`numpy.array` of `float`, shape (`shape` [1],): Cell size along the v-axis.

property `z_cell_delimiters`: `np.ndarray` | `None`

`numpy.array` of `float`: Nodal offsets along the z-axis relative to the origin (positive up).

property z_cells: `np.ndarray` | `None`

`numpy.array` of `float`, shape (*shape* [2],): Cell size along the z-axis

geoh5py.objects.curve

class `geoh5py.objects.curve.Curve`(*object_type*: `ObjectType`, *name*='Curve', ***kwargs*)

Bases: `CellObject`

Curve object defined by a series of line segments (*cells*) connecting *vertices*.

property cells: `np.ndarray` | `None`

`numpy.ndarray` of `int`, shape (*, 2): Array of indices defining segments connecting vertices. Defined based on *parts* if set by the user.

property current_line_id: `uuid.UUID` | `None`

classmethod `default_type_uid()` → `UUID`

Returns

Default unique identifier

property parts

`numpy.array` of `int`, shape (*n_vertices*, 2): Group identifiers for vertices connected by line segments as defined by the *cells* property. The definition of the *cells* property get modified by the setting of parts.

property unique_parts

list of `int`: Unique *parts* identifiers.

geoh5py.objects.drape_model

class `geoh5py.objects.drape_model.DrapeModel`(*object_type*: `ObjectType`, ***kwargs*)

Bases: `ObjectBase`

Drape (curtain) model object made up of layers and prisms.

property centroids

`numpy.array` of `float`, shape (*n_cells*, 3): Cell center locations in world coordinates.

```
centroids = [
    [x_1, y_1, z_1],
    ...,
    [x_N, y_N, z_N]
]
```

clip_by_extent(*bounds*: `np.ndarray`) → `ObjectBase` | `None`

Find indices of cells within a rectangular bounds.

Parameters

- **bounds** – shape(2, 2) Bounding box defined by the South-West and North-East coordinates. Extents can also be provided as 3D coordinates with shape(2, 3) defining the top and bottom limits.
- **attributes** – Dictionary of attributes to clip by extent.

classmethod `default_type_uid()` → `UUID`

property layers: `np.ndarray` | `None`
 layers

property n_cells
 int: Number of cells.

property prisms: `np.ndarray` | `None`
 prisms

geoh5py.objects.drillhole

class `geoh5py.objects.drillhole.Drillhole`(*object_type*: `ObjectType`, ***kwargs*)

Bases: `Points`

Drillhole object class defined by

Warning: Not yet implemented.

add_data(*data*: `dict`, *property_group*: `str` | `PropertyGroup` | `None` = `None`, *collocation_distance*=`None`) → `Data` | `list[Data]`

Create `Data` specific to the drillhole object from dictionary of name and arguments. A keyword ‘depth’ or ‘from-to’ with corresponding depth values is expected in order to locate the data along the well path.

Parameters

data – Dictionary of data to be added to the object, e.g.

```
data_dict = {
    "data_A": {
        'values', [v_1, v_2, ...],
        "from-to": numpy.ndarray,
    },
    "data_B": {
        'values', [v_1, v_2, ...],
        "depth": numpy.ndarray,
    },
}
```

Parameters

property_group – Name or `PropertyGroup` used to group the data.

Returns

List of new `Data` objects.

add_vertices(*xyz*)

Function to add vertices to the drillhole

property cells: `np.ndarray` | `None`

`numpy.ndarray` of `int`, shape `(*, 2)`: Array of indices defining segments connecting vertices.

property collar

`numpy.array` of `float`, shape `(3,)`: Coordinates of the collar

property cost

float: Cost estimate of the drillhole

property default_collocation_distance

Minimum collocation distance for matching depth on merge

classmethod default_type_uid() → UUID**property depths:** *FloatData* | None**desurvey**(*depths*)

Function to return x, y, z coordinates from depth.

property end_of_hole: float | None

End of drillhole in meters

property from_

Depth data corresponding to the tops of the interval values.

property locations: np.ndarray | None

Lookup array of the well path in x, y, z coordinates.

property planning: str

Status of the hole on of “Default”, “Ongoing”, “Planned”, “Completed” or “No status”

sort_depths()

Read the ‘DEPTH’ data and sort all Data.values if needed

property surveys: ndarray

Coordinates of the surveys

property to_

Depth data corresponding to the bottoms of the interval values.

property trace: np.ndarray | None

numpy.array: Drillhole trace defining the path in 3D

property trace_depth: np.ndarray | None

numpy.array: Drillhole trace depth from top to bottom

validate_data(*attributes: dict, property_group=None, collocation_distance=None*) → tuple

Validate input drillhole data attributes.

Parameters

- **attributes** – Dictionary of data attributes.
- **property_group** – Input property group to validate against.

validate_interval_data(*from_to: np.ndarray | list, values: np.ndarray, collocation_distance: float = 0.0001*)

Compare new and current depth values, append new vertices if necessary and return an augmented values vector that matches the vertices indexing.

validate_log_data(*depth: ndarray, input_values: ndarray, collocation_distance=0.0001*) → ndarray

Compare new and current depth values. Append new vertices if necessary and return an augmented values vector that matches the vertices indexing.

`geoh5py.objects.drillhole.compute_deviation(surveys: ndarray) → ndarray`

Compute deviation distances from survey parameters.

Parameters

surveys – Array of azimuth, dip and depth values.

`geoh5py.objects.drillhole.deviation_x(azimuth, dip)`

Compute the easting deviation.

Parameters

- **azimuth** – Degree angle clockwise from North
- **dip** – Degree angle positive down from horizontal

Return deviation

Change in easting distance.

`geoh5py.objects.drillhole.deviation_y(azimuth, dip)`

Compute the northing deviation.

Parameters

- **azimuth** – Degree angle clockwise from North
- **dip** – Degree angle positive down from horizontal

Return deviation

Change in northing distance.

`geoh5py.objects.drillhole.deviation_z(_, dip)`

Compute the vertical deviation.

Parameters

dip – Degree angle positive down from horizontal

Return deviation

Change in vertical distance.

geoh5py.objects.geo_image

`class geoh5py.objects.geo_image.GeoImage(object_type: ObjectType, **kwargs)`

Bases: [ObjectBase](#)

Image object class.

Warning: Not yet implemented.

property cells: np.ndarray | None

numpy.ndarray of int, shape (*, 2): Array of indices defining segments connecting vertices. Defined based on [parts](#) if set by the user.

classmethod default_type_uid() → UUID

property default_vertices

Assign the default vertices based on image pixel count

georeference(*reference*: *np.ndarray* | *list*, *locations*: *np.ndarray* | *list*)

Georeference the image vertices (corners) based on input reference and corresponding world coordinates.

Parameters

- **reference** – Array of integers representing the reference used as reference points.
- **locations** – Array of floats for the corresponding world coordinates for each input pixel.

Return vertices

Corners (vertices) in world coordinates.

georeferencing_from_tiff()

Get the geographic information from the PIL Image to georeference it. Run the `georeference()` method of the object.

property image

Get the image as a `PIL.Image` object.

property image_data

Get the `FilenameData` entity holding the image.

property image_georeferenced: `Image.Image` | `None`

Get the image as a georeferenced `PIL.Image` object.

save_as(*name*: *str*, *path*: *str* | *Path* = "")

Function to save the geoimage into an image file. If the name ends by `‘.tif’` or `‘.tiff’` and the tag is not `None` then the image is saved as georeferenced tiff image ; else, the image is save with `PIL.Image`’s save function.
:param name: the name to give to the image. :param path: the path of the file of the image, default: `‘.’`.

set_tag_from_vertices()

If tag is `None`, set the basic tag values based on vertices in order to export as a georeferenced `.tiff`. WARNING: this function must be used after `georeference()`.

property tag: `dict` | `None`

Georeferencing information of a tiff image stored in the header. :return: a dictionary containing the `PIL.Image.tag` information.

to_grid2d(*transform*: *str* = `‘GRAY’`, ***grid2d_kwargs*) → *Grid2D*

Create a `geoh5py.objects.grid2d.Grid2D` from the geoimage in the same workspace. :param transform: the type of transform ; if `“GRAY”` convert the image to grayscale ; if `“RGB”` every band is sent to a data of a grid. :return: the new created *geoh5py.objects.grid2d.Grid2D*.

property vertices: `np.ndarray` | `None`

vertices: Defines the four corners of the `geo_image`

geoh5py.objects.grid2d

class `geoh5py.objects.grid2d.Grid2D`(*object_type*: *ObjectType*, ***kwargs*)

Bases: *ObjectBase*

Rectilinear 2D grid of uniform cell size. The grid can be oriented in 3D space through horizontal *rotation* and *dip* parameters. Nodal coordinates are determined relative to the origin and the sign of cell delimiters.

property cell_center_u: `np.ndarray` | `None`

`numpy.array` of float, shape(*u_count*,): Cell center local coordinate along the u-axis.

property cell_center_v: np.ndarray | None

numpy.array of float shape([u_count](#),): The cell center local coordinate along the v-axis.

property centroids: np.ndarray | None

numpy.array of float, shape ([n_cells](#), 3): Cell center locations in world coordinates.

```
centroids = [  
    [x_1, y_1, z_1],  
    ...,  
    [x_N, y_N, z_N]  
]
```

classmethod default_type_uid() → UUID

Returns

Default unique identifier

property dip: float

float: Dip angle from horizontal (positive down) in degrees.

property n_cells: int | None

int: Total number of cells.

property origin: ndarray

numpy.array of float, shape (3,): Coordinates of the origin.

property rotation: float

float: Clockwise rotation angle (degree) about the vertical axis.

property shape: tuple | None

list of int, len (2,): Number of cells along the u and v-axis.

to_geoimage(keys: list | str, **geoimage_kwargs) → objects.GeoImage

Create a :obj:geoh5py.objects.geo_image.GeoImage object from the current Grid2D. :param keys: the list of the data name to pass as band in the image. Warning: The len of the list can only be 1, 3, 4 (Pillow restrictions). :return: a new georeferenced [geoh5py.objects.geo_image.GeoImage](#).

property u_cell_size: np.ndarray | None

np.ndarray: Cell size along the u-axis.

property u_count: int | None

int: Number of cells along u-axis

property v_cell_size: np.ndarray | None

np.ndarray: Cell size along the v-axis

property v_count: int | None

int: Number of cells along v-axis

property vertical: bool | None

bool: Set the grid to be vertical.

geoh5py.objects.integrator

class geoh5py.objects.integrator.**IntegratorPoints**(*object_type*: [ObjectType](#), ***kwargs*)

Bases: [Points](#)

INTEGRATOR Points object. Sub-class of [geoh5py.objects.points.Points](#).

classmethod **default_type_uid**() → UUID

class geoh5py.objects.integrator.**NeighbourhoodSurface**(*object_type*: [ObjectType](#), ***kwargs*)

Bases: [Surface](#)

Points object made up of vertices.

classmethod **default_type_uid**() → UUID

Default type uid.

geoh5py.objects.label

class geoh5py.objects.label.**Label**(*object_type*: [ObjectType](#), ***kwargs*)

Bases: [ObjectBase](#)

Label object for annotation in viewport.

Warning: Not yet implemented.

classmethod **default_type_uid**() → UUID

geoh5py.objects.notype_object

class geoh5py.objects.notype_object.**NoTypeObject**(*object_type*: [ObjectType](#), ***kwargs*)

Bases: [ObjectBase](#)

Generic Data object without a registered type

classmethod **default_type_uid**() → UUID

geoh5py.objects.object_base

class geoh5py.objects.object_base.**ObjectBase**(*object_type*: [ObjectType](#), ***kwargs*)

Bases: [Entity](#)

Object base class.

add_comment(*comment*: *str*, *author*: *str* | *None* = *None*)

Add text comment to an object.

Parameters

- **comment** – Text to be added as comment.
- **author** – Name of author or defaults to [contributors](#).

add_data(data: dict, property_group: str | [PropertyGroup](#) | None = None) → [Data](#) | list[[Data](#)]

Create [Data](#) from dictionary of name and arguments. The provided arguments can be any property of the target [Data](#) class.

Parameters

data – Dictionary of data to be added to the object, e.g.

```
data = {
    "data_A": {
        'values': [v_1, v_2, ...],
        'association': 'VERTEX'
    },
    "data_B": {
        'values': [v_1, v_2, ...],
        'association': 'CELLS'
    },
}
```

Returns

List of new [Data](#) objects.

add_data_to_group(data: list | [Data](#) | [uuid.UUID](#), property_group: str | [PropertyGroup](#)) → [PropertyGroup](#)

Append data children to a [PropertyGroup](#). All given data must be children of the parent object.

Parameters

- **data** – [Data](#) object, [uid](#) or [name](#) of data.
- **property_group** – Name or [PropertyGroup](#). A new group is created if none exist with the given name.

Returns

The target property group.

property cells

`numpy.array of int`: Array of indices defining the connection between [vertices](#).

clip_by_extent(bounds: `np.ndarray`) → [ObjectBase](#) | None

Find indices of cells within a rectangular bounds.

Parameters

- **bounds** – `shape(2, 2)` Bounding box defined by the South-West and North-East coordinates. Extents can also be provided as 3D coordinates with `shape(2, 3)` defining the top and bottom limits.
- **attributes** – Dictionary of attributes to clip by extent.

property comments

Fetch a [CommentsData](#) entity from children.

property converter

Returns

The converter for the object.

copy_from_extent(bounds: `np.ndarray`, parent=None, copy_children: `bool = True`) → [ObjectBase](#) | None

Find indices of vertices within a rectangular bounds.

Parameters

- **bounds** – shape(2, 2) Bounding box defined by the South-West and North-East coordinates. Extents can also be provided as 3D coordinates with shape(2, 3) defining the top and bottom limits.
- **attributes** – Dictionary of attributes to clip by extent.

abstract classmethod default_type_uid() → UUID

property entity_type: *ObjectType*

EntityType: Object type.

property extent

property faces

find_or_create_property_group(kwargs)** → *PropertyGroup*

Find or create *PropertyGroup* from given name and properties.

Parameters

kwargs – Any arguments taken by the *PropertyGroup* class.

Returns

A new or existing *PropertyGroup*

classmethod find_or_create_type(workspace: workspace.Workspace, **kwargs) → *ObjectType*

Find or create a type instance for a given object class.

Parameters

workspace – Target *Workspace*.

Returns

The *ObjectType* instance for the given object class.

get_data(name: str | uuid.UUID) → list[*Data*]

Get a child *Data* by name.

Parameters

name – Name of the target child data

Returns

A list of children *Data* objects

get_data_list(attribute='name') → list[str]

Get a list of names of all children *Data*.

Returns

List of names of data associated with the object.

property last_focus: str

bool: Object visible in camera on start.

property n_cells: int | None

int: Number of cells.

property n_vertices: int | None

int: Number of vertices.

property property_groups: list[*PropertyGroup*] | None

List of *PropertyGroup*.

remove_children_values(indices: list[int], association: str)

remove_property_groups(*property_groups*: [PropertyGroup](#) | *list*[[PropertyGroup](#)])

validate_data_association(*attribute_dict*)

Get a dictionary of attributes and validate the data 'association' keyword.

static validate_data_type(*attribute_dict*)

Get a dictionary of attributes and validate the type of data.

property vertices

numpy.array of float, shape (*, 3): Array of x, y, z coordinates defining the position of points in 3D space.

geoh5py.objects.object_type

class geoh5py.objects.object_type.**ObjectType**(*workspace*: [workspace.Workspace](#), ***kwargs*)

Bases: [EntityType](#)

Object type class

static create_custom(*workspace*: [workspace.Workspace](#)) → [ObjectType](#)

Creates a new instance of ObjectType for an unlisted custom Object type with a new auto-generated UUID.

Parameters

workspace – An active Workspace class

classmethod find_or_create(*workspace*: [workspace.Workspace](#), *entity_class*, ***kwargs*) → [ObjectType](#)

Find or creates an EntityType with given uuid.UUID that matches the given Group implementation class.

It is expected to have a single instance of EntityType in the Workspace for each concrete Entity class.

Parameters

- **workspace** – An active Workspace class
- **entity_class** – An Group implementation class.

Returns

A new instance of GroupType.

geoh5py.objects.octree

class geoh5py.objects.octree.**Octree**(*object_type*: [ObjectType](#), ***kwargs*)

Bases: [ObjectBase](#)

Octree mesh class that uses a tree structure such that cells can be subdivided it into eight octants.

base_refine()

Refine the mesh to its base octree level resulting in a single cell along the shortest dimension.

property centroids

numpy.array of float, shape ([n_cells](#), 3): Cell center locations in world coordinates.

```
centroids = [  
    [x_1, y_1, z_1],  
    ...,  
    [x_N, y_N, z_N]  
]
```

classmethod `default_type_uid()` → UUID

property `n_cells: int | None`

int: Total number of cells in the mesh

property `octree_cells: np.ndarray | None`

numpy.ndarray of int, shape (`n_cells`, 4): Array defining the i, j, k position and size of each cell. The size defines the width of a cell in number of base cells.

```
cells = [
    [i_1, j_1, k_1, size_1],
    ...,
    [i_N, j_N, k_N, size_N]
]
```

property `origin`

numpy.array of float, shape (3,): Coordinates of the origin

property `rotation: float`

float: Clockwise rotation angle (degree) about the vertical axis.

property `shape: tuple | None`

list of int, len (3,): Number of cells along the u, v and w-axis.

property `u_cell_size: float | None`

float: Base cell size along the u-axis.

property `u_count: int | None`

int: Number of cells along u-axis.

property `v_cell_size: float | None`

float: Base cell size along the v-axis.

property `v_count: int | None`

int: Number of cells along v-axis.

property `w_cell_size: float | None`

float: Base cell size along the w-axis.

property `w_count: int | None`

int: Number of cells along w-axis.

geoh5py.objects.points

class `geoh5py.objects.points.Points(object_type: ObjectType, name='Points', **kwargs)`

Bases: *ObjectBase*

Points object made up of vertices.

clip_by_extent(*bounds: np.ndarray*) → *Points* | None

Find indices of vertices within a rectangular bounds.

Parameters

bounds – shape(2, 2) Bounding box defined by the South-West and North-East coordinates. Extents can also be provided as 3D coordinates with shape(2, 3) defining the top and bottom limits.

```
classmethod default_type_uid() → UUID  
  
remove_vertices(indices: list[int])  
    Safely remove vertices and corresponding data entries.  
  
property vertices: np.ndarray | None  
    vertices
```

geoh5py.objects.surface

```
class geoh5py.objects.surface.Surface(object_type: ObjectType, **kwargs)  
    Bases: CellObject  
    Surface object defined by vertices and cells  
  
    property cells: np.ndarray | None  
        Array of vertices index forming triangles :return cells: numpy.array of int, shape ("*", 3)  
  
    classmethod default_type_uid() → UUID  
        Default type uid.
```

2.3.5 geoh5py.shared

geoh5py.shared.concatenation

```
class geoh5py.shared.concatenation.Concatenated(entity_type, **kwargs)  
    Bases: Entity  
    Base class modifier for concatenated objects and data.  
  
    property concat_attr_str: str  
        String identifier for the concatenated attributes.  
  
    property concatenator: Concatenator  
        Parental Concatenator entity.  
  
class geoh5py.shared.concatenation.ConcatenatedData(entity_type, **kwargs)  
    Bases: Concatenated  
    property parent: ConcatenatedObject  
  
    property property_group  
        Get the property group containing the data interval.  
  
class geoh5py.shared.concatenation.ConcatenatedDrillhole(entity_type, **kwargs)  
    Bases: ConcatenatedObject  
    property from_: list[Data]  
        Depth data corresponding to the tops of the interval values.  
  
    sort_depths()  
        Bypass sort_depths from previous version.  
  
    property to_: list[Data]  
        Depth data corresponding to the bottoms of the interval values.
```


validate_data(*attributes: dict, property_group=None, collocation_distance=None*) → tuple

Validate input drillhole data attributes.

Parameters

- **attributes** – Dictionary of data attributes.
- **property_group** – Input property group to validate against.

validate_interval_data(*name: str | None, from_to: list | np.ndarray | None, values: np.ndarray, property_group: str | ConcatenatedPropertyGroup | None = None, collocation_distance=0.0001*) → *ConcatenatedPropertyGroup*

Compare new and current depth values and re-use the property group if possible. Otherwise a new property group is added.

Parameters

- **from_to** – Array of from-to values.
- **values** – Data values to be added on the from-to intervals.
- **property_group** – Property group name

Collocation_distance

Threshold on the comparison between existing depth values.

class geoh5py.shared.concatenation.**ConcatenatedObject**(*entity_type, **kwargs*)

Bases: *Concatenated, ObjectBase*

find_or_create_property_group(***kwargs*) → *ConcatenatedPropertyGroup*

Find or create *PropertyGroup* from given name and properties.

Parameters

kwargs – Any arguments taken by the *PropertyGroup* class.

Returns

A new or existing *PropertyGroup*

get_data(*name: str | uuid.UUID*) → list[*Data*]

Generic function to get data values from object.

get_data_list(*attribute='name'*)

Get list of data names.

property parent: *Concatenator*

property property_groups: list | None

List of *PropertyGroup*.

class geoh5py.shared.concatenation.**ConcatenatedPropertyGroup**(*parent: ConcatenatedObject, **kwargs*)

Bases: *PropertyGroup*

property from_

Return the data entities definind the ‘from’ depth intervals.

property parent

The parent *ObjectBase*

property to_

Return the data entities definind the ‘to’ depth intervals.

class geoh5py.shared.concatenation.**Concatenator**(*group_type*: [GroupType](#), ***kwargs*)

Bases: [Group](#)

Class modifier for concatenation of objects and data.

add_save_concatenated(*child*) → None

Add or save a concatenated entity.

Parameters

child – Concatenated entity

property attributes_keys: list | None

List of uuids present in the concatenated attributes.

property concat_attr_str: str

String identifier for the concatenated attributes.

property concatenated_attributes: dict | None

Dictionary of concatenated objects and data attributes.

property concatenated_object_ids: list[bytes] | None

Dictionary of concatenated objects and data concatenated_object_ids.

copy(*parent*=None, *copy_children*: bool = True, ***kwargs*)

Function to copy an entity to a different parent entity.

Parameters

- **parent** – Target parent to copy the entity under. Copied to current [parent](#) if None.
- **copy_children** – Create copies of all children entities along with it.

Return entity

Registered Entity to the workspace.

property data: dict

Concatenated data values stored as a dictionary.

delete_index_data(*label*: str, *index*: int) → None

fetch_concatenated_data_index()

Extract concatenation arrays.

fetch_concatenated_objects() → dict

Load all concatenated children.

fetch_index(*entity*: [Concatenated](#), *field*: str) → int | None

Fetch the array index for specific concatenated object and data field.

Parameters

- **entity** – Parent entity with data
- **field** – Name of the target data.

fetch_start_index(*entity*: [Concatenated](#), *label*: str) → int

Fetch starting index for a given entity and label. Existing date is removed such that new entries can be appended.

Parameters

- **entity** – Concatenated entity to be added.

- **label** – Name of the attribute requiring an update.

fetch_values(*entity*: [Concatenated](#), *field*: *str*) → np.ndarray | None

Get an array of values from concatenated data.

Parameters

- **entity** – Parent entity with data
- **field** – Name of the target data.

get_concatenated_attributes(*uid*: *bytes* | *str* | *uuid.UUID*) → dict

Fast reference index to concatenated attribute keys.

property index: dict

Concatenated index stored as a dictionary.

property property_group_ids: list | None

Dictionary of concatenated objects and data property_group_ids.

remove_entity(*entity*: [Concatenated](#))

Remove a concatenated entity.

save_attribute(*field*: *str*)

Save a concatenated attribute.

Parameters

field – Name of the attribute

update_array_attribute(*entity*: [Concatenated](#), *field*: *str*, *remove=False*) → None

Update values stored as data. Row data and indices are first remove then appended.

Parameters

- **entity** – Concatenated entity with array values.
- **field** – Name of the valued field.

update_attributes(*entity*: [Concatenated](#), *label*: *str*) → None

Update a concatenated entity.

update_concatenated_attributes(*entity*: [Concatenated](#)) → None

Update the concatenated attributes. :param entity: Concatenated entity with attributes.

geoh5py.shared.entity

class geoh5py.shared.entity.**Entity**(*uid*: *uuid.UUID* | *None* = *None*, *name*='Entity', ***kwargs*)

Bases: ABC

Base Entity class

add_children(*children*: list[*shared.Entity*])

Parameters

children – Add a list of entities as [children](#)

add_file(*file*: *str*)

Add a file to the object or group stored as bytes on a FilenameData

Parameters

file – File name with path to import.

property allow_delete: bool

bool Entity can be deleted from the workspace.

property allow_move: bool

bool Entity can change *parent*

property allow_rename: bool

bool Entity can change name

property attribute_map: dict

dict Correspondence map between property names used in geoh5py and geoh5.

property children

list Children entities in the workspace tree

property clipping_ids: list[uuid.UUID] | None

List of clipping uuids

copy(*parent=None, copy_children: bool = True, **kwargs*)

Function to copy an entity to a different parent entity.

Parameters

- **parent** – Target parent to copy the entity under. Copied to current *parent* if None.
- **copy_children** – (Optional) Create copies of all children entities along with it.
- **kwargs** – Additional keyword arguments to pass to the copy constructor.

Return entity

Registered Entity to the workspace.

classmethod create(*workspace, **kwargs*)

Function to create an entity.

Parameters

- **workspace** – Workspace to be added to.
- **kwargs** – List of keyword arguments defining the properties of a class.

Return entity

Registered Entity to the workspace.

abstract property entity_type: shared.EntityType

classmethod fix_up_name(*name: str*) → str

If the given name is not a valid one, transforms it to make it valid :return: a valid name built from the given name. It simply returns the given name if it was already valid.

get_entity(*name: str | uuid.UUID*) → list[*Entity*]

Get a child *Data* by name.

Parameters

- **name** – Name of the target child data
- **entity_type** – Sub-select entities based on type.

Returns

A list of children Data objects

get_entity_list(*entity_type*=<class 'abc.ABC'>) → list[str]

Get a list of names of all children *Data*.

Parameters

entity_type – Option to sub-select based on type.

Returns

List of names of data associated with the object.

property metadata: dict | None

Metadata attached to the entity.

property name: str

str Name of the entity

property on_file: bool

Whether this Entity is already stored on *h5file*.

property parent

property partially_hidden: bool

Whether this Entity is partially hidden.

property public: bool

Whether this Entity is accessible in the workspace tree and other parts of the the user interface in ANALYST.

reference_to_uid(*value*: *Entity* | *str* | *uuid.UUID*) → list[*uuid.UUID*]

General entity reference translation.

Parameters

value – Either an *Entity*, string or uuid

Returns

List of unique identifier associated with the input reference.

remove_children(*children*: list[*shared.Entity*])

Remove children from the list of children entities.

Parameters

children – List of entities

Warning: Removing a child entity without re-assigning it to a different parent may cause it to become inactive. Inactive entities are removed from the workspace by *remove_none_referents()*.

remove_data_from_group(*data*: list | *Entity* | *uuid.UUID* | *str*, *name*: *str* | *None* = *None*) → *None*

Remove data children to a *PropertyGroup* All given data must be children of the parent object.

Parameters

- **data** – *Data* object, *uid* or *name* of data.
- **name** – Name of a *PropertyGroup*. A new group is created if none exist with the given name.

save(*add_children*: *bool* = *True*)

Alias method of *save_entity()*.

Parameters

add_children – Option to also save the children.

property uid: UUID

property visible: bool

Whether the Entity is visible in camera (checked in ANALYST object tree).

property workspace: *Workspace*

Workspace to which the Entity belongs to.

geoh5py.shared.entity_type

```
class geoh5py.shared.entity_type.EntityType(workspace: ws.Workspace, uid: uuid.UUID | None = None,
                                             **kwargs)
```

Bases: ABC

property attribute_map

dict Correspondence map between property names used in geoh5py and geoh5.

property description: str | None

classmethod find(workspace: ws.Workspace, type_uid: uuid.UUID) → EntityTypeT | None

Finds in the given Workspace the EntityType with the given UUID for this specific EntityType implementation class.

Returns

EntityType of None

property name: str | None

property on_file: bool

bool Entity already present in *h5file*.

property uid: UUID

uuid.UUID The unique identifier of an entity, either as stored in geoh5 or generated in uuid4() format.

property workspace: ws.Workspace

Workspace registering this type.

geoh5py.shared.exceptions

```
exception geoh5py.shared.exceptions.AssociationValidationError(name: str, value: Entity |
                                                                PropertyGroup | UUID,
                                                                validation: Entity | Workspace)
```

Bases: *BaseValidationError*

Error on association between child and parent entity validation.

static message(name, value, validation)

Builds custom error message.

```
exception geoh5py.shared.exceptions.AtLeastOneValidationError(name: str, value: list[str])
```

Bases: *BaseValidationError*

static message(*name, value, validation=None*)

Builds custom error message.

exception `geoh5py.shared.exceptions.BaseValidationError`

Bases: `ABC, Exception`

Base class for custom exceptions.

abstract static message(*name, value, validation*)

Builds custom error message.

exception `geoh5py.shared.exceptions.Geoh5FileClosedError`

Bases: `ABC, Exception`

Error for closed geoh5 file.

exception `geoh5py.shared.exceptions.JSONParameterValidationError`(*name: str, err: str*)

Bases: `Exception`

Error on uuid validation.

static message(*name, err*)

exception `geoh5py.shared.exceptions.OptionalValidationError`(*name: str, value: Any | None, validation: bool*)

Bases: `BaseValidationError`

Error if None value provided to non-optional parameter.

static message(*name, value, validation*)

Builds custom error message.

exception `geoh5py.shared.exceptions.PropertyGroupValidationError`(*name: str, value: PropertyGroup, validation: str*)

Bases: `BaseValidationError`

Error on property group validation.

static message(*name, value, validation*)

Builds custom error message.

exception `geoh5py.shared.exceptions.RequiredValidationError`(*name: str*)

Bases: `BaseValidationError`

static message(*name, value=None, validation=None*)

Builds custom error message.

exception `geoh5py.shared.exceptions.ShapeValidationError`(*name: str, value: tuple[int], validation: tuple[int] | str*)

Bases: `BaseValidationError`

Error on shape validation.

static message(*name, value, validation*)

Builds custom error message.

exception `geoh5py.shared.exceptions.TypeValidationError`(*name: str, value: str, validation: str | list[str]*)

Bases: `BaseValidationError`

Error on type validation.

static message(*name*, *value*, *validation*)

Builds custom error message.

exception `geoh5py.shared.exceptions.UUIDValidationError`(*name*: *str*, *value*: *str*)

Bases: `BaseValidationError`

Error on uuid string validation.

static message(*name*, *value*, *validation*=None)

Builds custom error message.

exception `geoh5py.shared.exceptions.ValueValidationError`(*name*: *str*, *value*: *Any*, *validation*:
list[*Any*])

Bases: `BaseValidationError`

Error on value validation.

static message(*name*, *value*, *validation*)

Builds custom error message.

geoh5py.shared.utils

`geoh5py.shared.utils.as_str_if_utf8_bytes`(*value*) → *str*

Convert bytes to string

`geoh5py.shared.utils.as_str_if_uuid`(*value*: *UUID* | *Any*) → *str* | *Any*

Convert UUID to string used in geoh5.

`geoh5py.shared.utils.bool_value`(*value*: *int8*) → *bool*

Convert logical int8 to bool.

`geoh5py.shared.utils.compare_entities`(*object_a*, *object_b*, *ignore*: *list* | *None* = *None*, *decimal*: *int* = 6)
→ *None*

`geoh5py.shared.utils.dict_mapper`(*val*, *string_funcs*: *list*[*Callable*], **args*, *omit*: *dict* | *None* = *None*) → *dict*
Recursion through nested dictionaries and applies mapping functions to values.

Parameters

- **val** – Value (could be another dictionary) to apply transform functions.
- **string_funcs** – Functions to apply on values within the input dictionary.
- **omit** – Dictionary of functions to omit.

Return val

Transformed values

`geoh5py.shared.utils.entity2uuid`(*value*: *Any*) → *UUID* | *Any*

Convert an entity to its UUID.

`geoh5py.shared.utils.fetch_h5_handle`(*file*: *str* | *h5py.File* | *Path*, *mode*: *str* = 'r') → *h5py.File*

Open in read+ mode a geoh5 file from string. If receiving a file instead of a string, merely return the given file.

Parameters

- **file** – Name or handle to a geoh5 file.
- **mode** – Set the h5 read/write mode

Return h5py.File

Handle to an opened h5py file.

`geoh5py.shared.utils.get_attributes(entity, omit_list=(), attributes=None)`

Extract the attributes of an object with omissions.

`geoh5py.shared.utils.is_uuid(value: str) → bool`

Check if a string is UUID compliant.

`geoh5py.shared.utils.iterable(value: Any, checklen: bool = False) → bool`

Checks if object is iterable.

Parameters**value**

[Object to check for iterableness.]

checklen

[Restrict objects with `__iter__` method to `len > 1`.]

Returns

True if object has `__iter__` attribute but is not string or dict type.

`geoh5py.shared.utils.iterable_message(valid: list[Any] | None) → str`

Append possibly iterable valid: “Must be (one of): {valid}.”.

`geoh5py.shared.utils.mask_by_extent(locations: np.ndarray, extent: np.ndarray | list[list]) → np.ndarray`

Find indices of locations within a rectangular extent.

Parameters

- **locations** – `shape(*, 3)` Coordinates to be evaluated.
- **extent** – `shape(2, 2)` Limits defined by the South-West and North-East corners. Extents can also be provided as 3D coordinates with `shape(2, 3)` defining the top and bottom limits.

`geoh5py.shared.utils.match_values(vec_a, vec_b, collocation_distance=0.0001) → ndarray`

Find indices of matching values between two arrays, within `collocation_distance`.

Param

`vec_a`, list or `numpy.ndarray` Input sorted values

Param

`vec_b`, list or `numpy.ndarray` Query values

Returns

indices, `numpy.ndarray` Pairs of indices for matching values between the two arrays such that `vec_a[ind[:, 0]] == vec_b[ind[:, 1]]`.

`geoh5py.shared.utils.merge_arrays(head, tail, replace='A->B', mapping=None, collocation_distance=0.0001, return_mapping=False) → ndarray`

Given two `numpy.array`s of different length, find the matching values and append both arrays.

Param

`head`, `numpy.array` of float First vector of `shape(M,)` to be appended.

Param

`tail`, `numpy.array` of float Second vector of `shape(N,)` to be appended

Param

`mapping=None`, `numpy.ndarray` of int Optional array where values from the head are replaced by the tail.

Param

collocation_distance=1e-4, float Tolerance between matching values.

Returns

numpy.array shape(O,) Unique values from head to tail without repeats, within collocation_distance.

geoh5py.shared.utils.**overwrite_kwargs**(to_overwrite: dict, kwargs_to_add: dict) → dict

Overwrite kwargs with overwrite. :param to_overwrite: Dictionary of kwargs to overwrite. :param kwargs_to_add: Dictionary of kwargs to modify to_overwrite.

geoh5py.shared.utils.**str2uuid**(value: Any) → UUID | Any

Convert string to UUID

geoh5py.shared.utils.**uuid2entity**(value: UUID, workspace: Workspace) → Entity | Any

Convert UUID to a known entity.

geoh5py.shared.validators

class geoh5py.shared.validators.**AssociationValidator**(**kwargs)

Bases: [BaseValidator](#)

Validate the association between data and parent object.

classmethod **validate**(name: str, value: Entity | PropertyGroup | UUID | None, valid: Entity | Workspace) → None

Parameters

- **name** – Parameter identifier.
- **value** – Input parameter value.
- **valid** – Expected value shape

validator_type = 'association'

class geoh5py.shared.validators.**AtLeastOneValidator**(**kwargs)

Bases: [BaseValidator](#)

classmethod **validate**(name, value, valid)

Custom validation function.

validator_type = 'one_of'

class geoh5py.shared.validators.**BaseValidator**(**kwargs)

Bases: ABC

Concrete base class for validators.

abstract classmethod **validate**(name: str, value: Any, valid: Any)

Custom validation function.

abstract property **validator_type**: str

Validation type identifier.

class geoh5py.shared.validators.**OptionalValidator**(**kwargs)

Bases: [BaseValidator](#)

Validate that forms contain optional parameter if None value is given.

classmethod `validate(name: str, value: Any | None, valid: bool) → None`

Parameters

- **name** – Parameter identifier.
- **value** – Input parameter value.
- **valid** – True if optional keyword in form for parameter.

validator_type = 'optional'

class `geoh5py.shared.validators.PropertyGroupValidator(**kwargs)`

Bases: `BaseValidator`

Validate property_group from parent entity.

classmethod `validate(name: str, value: PropertyGroup, valid: str) → None`

Custom validation function.

validator_type = 'property_group_type'

class `geoh5py.shared.validators.RequiredValidator(**kwargs)`

Bases: `BaseValidator`

Validate that required keys are present in parameter.

classmethod `validate(name: str, value: Any, valid: bool) → None`

Parameters

- **name** – Parameter identifier.
- **value** – Input parameter value.
- **valid** – Assert to be required

validator_type = 'required'

class `geoh5py.shared.validators.ShapeValidator(**kwargs)`

Bases: `BaseValidator`

Validate the shape of provided value.

classmethod `validate(name: str, value: Any, valid: tuple[int]) → None`

Parameters

- **name** – Parameter identifier.
- **value** – Input parameter value.
- **valid** – Expected value shape

validator_type = 'shape'

class `geoh5py.shared.validators.TypeValidator(**kwargs)`

Bases: `BaseValidator`

Validate the value type from a list of valid types.

```
classmethod validate(name: str, value: Any, valid: list[type] | type) → None
```

Parameters

- **name** – Parameter identifier.
- **value** – Input parameter value.
- **valid** – List of accepted value types

```
validator_type = 'types'
```

```
class geoh5py.shared.validators.UUIDValidator(**kwargs)
```

Bases: [BaseValidator](#)

Validate a uuui.UUID value or uuid string.

```
classmethod validate(name: str, value: Any, valid: None = None) → None
```

Parameters

- **name** – Parameter identifier.
- **value** – Input parameter uuid.
- **valid** – [Optional] Validate uuid from parental entity or known uuids

```
validator_type = 'uuid'
```

```
class geoh5py.shared.validators.ValueValidator(**kwargs)
```

Bases: [BaseValidator](#)

Validator that ensures that values are valid entries.

```
classmethod validate(name: str, value: Any, valid: list[float | str]) → None
```

Parameters

- **name** – Parameter identifier.
- **value** – Input parameter value.
- **valid** – List of accepted values

```
validator_type = 'values'
```

[geoh5py.shared.weakref_utils](#)

```
geoh5py.shared.weakref_utils.get_clean_ref(some_dict: dict[K, ReferenceType[T]], key: K) → T | None
```

Gets the referent value for the given **key** in a **some_dict** of **weakref** values. In case **key** points to a reference to a deleted value, remove that key from **some_dict** on the fly, and returns None.

Parameters

- **some_dict** – The dictionary of **weakref** values.
- **key** – The key

Returns

the referent value for **key** if found in the the dictionary, else None.

`geoh5py.shared.weakref_utils.insert_once(some_dict: dict[K, ReferenceType], key: K, value)`

Check if the reference to an Entity with uuid is already in use.

Parameters

- **some_dict** – Dictionary of UUID keys and weakref values.
- **key** – UUID key to be checked.
- **value** – Entity to be checked

Returns

Dictionary with clean weakref

`geoh5py.shared.weakref_utils.remove_none_referents(some_dict: dict[K, ReferenceType])`

Removes any key from the given `some_dict` where the value is a reference to a deleted value (that is where referent of the weakref value is None).

Parameters

some_dict – The dictionary to be cleaned up.

2.3.6 geoh5py.ui_json

`geoh5py.ui_json.constants`

`geoh5py.ui_json.input_file`

class `geoh5py.ui_json.input_file.InputFile`(*data: dict[str, Any] | None = None, ui_json: dict[str, Any] | None = None, validations: dict | None = None, validation_options: dict | None = None*)

Bases: `object`

Handles loading ui.json input files.

Attributes

data

[Input file content parsed to flat dictionary of key:value.]

ui_json: User interface serializable as ui.json format

workspace: Target :obj: `geoh5py.workspace.Workspace``

validations: Dictionary of validations for parameters in the input file

Methods

write_ui_json()	Writes a ui.json formatted file from 'data' attribute contents.
read_ui_json()	Reads a ui.json formatted file into 'data' attribute dictionary. Optionally filters ui.json fields other than 'value'.

association_validator = <geoh5py.shared.validators.AssociationValidator object>

property data

load(*input_dict: dict[str, Any]*)

Load data from dictionary and validate.

property name: str | None

Name of ui.json file.

classmethod numify(*ui_json: dict[str, Any]*) → dict[str, Any]

Convert inf, none and list strings to numerical types within a dictionary

Parameters

ui_json

dictionary containing ui.json keys, values, fields

Returns

Dictionary with inf, none and list string representations converted numerical types.

property path: str | None

Directory for the input/output ui.json file.

property path_name: str | None

static read_ui_json(*json_file: str, **kwargs*)

Read and create an InputFile from ui.json

property ui_json: dict | None

Dictionary representing the ui.json file with promoted values.

classmethod ui_validation(*ui_json: dict[str, Any]*)

Validation of the ui_json forms

update_ui_values(*data: dict, none_map=None*)

Update the ui.json values and enabled status from input data.

Parameters

- **data** – Key and value pairs expected by the ui_json.
- **none_map** – Map parameter 'None' values to non-null numeric types. The parameters in the dictionary are mapped to optional and disabled.

Raises

UserWarning – If attempting to set None value to non-optional parameter.

property validation_options

Pass validation options to the validators.

property validations: dict | None

property validators

property workspace

write_ui_json(*name: str | None = None, none_map: dict[str, Any] | None = None, path: str | None = None*)

Writes a formatted ui.json file from InputFile data

Parameters

- **name** – Name of the file
- **none_map** – Map parameter None values to non-null numeric types.
- **path** – Directory to write the ui.json to.

geoh5py.ui_json.templates

`geoh5py.ui_json.templates.bool_parameter`(*main: bool = True, label: str = 'Logical data', value: bool = False*) → dict

Checkbox for true/false choice.

Parameters

- **main** – Show ui in main.
- **label** – Label identifier.
- **value** – Input value.

Returns

Ui_json compliant dictionary.

`geoh5py.ui_json.templates.choice_string_parameter`(*main: bool = True, label: str = 'String data', choice_list: tuple = ('Option A', 'Option B'), value: str = 'Option A', optional: str | None = None*) → dict

Dropdown menu of string choices.

Parameters

- **main** – Show form in main.
- **label** – Label identifier.
- **value** – Input value.
- **choice_list** – List of options.
- **optional** – Make optional if not None. Initial state provided by not None value. Can be either 'enabled' or 'disabled'.

Returns

Ui_json compliant dictionary.

`geoh5py.ui_json.templates.data_parameter`(*main: bool = True, label: str = 'Data channel', association: str = 'Vertex', data_type: str = 'Float', data_group_type: str | None = None, parent: str = "", value: str = "", optional: str | None = None*) → dict

Dropdown menu of data from parental object.

Parameters

- **main** – Show form in main.
- **label** – Label identifier.
- **value** – Input value.
- **association** – Data association type from 'Vertex' or 'Cell'.
- **data_type** – Type of data selectable from 'Float', 'Integer' or 'Reference'.
- **data_group_type** – [Optional] Select from property_groups of type. '3D vector', 'Dip direction & dip', 'Strike & dip', or 'Multi-element'.
- **parent** – Parameter name corresponding to the parent object.
- **optional** – Make optional if not None. Initial state provided by not None value. Can be either 'enabled' or 'disabled'.

Returns

Ui_json compliant dictionary.

`geoh5py.ui_json.templates.data_value_parameter`(*main: bool = True, label: str = 'Data channel', association: str = 'Vertex', data_type: str = 'Float', parent: str = "", value: float = 0.0, is_value: bool = True, prop: UUID | Entity | None = None, optional: str | None = None*) → dict

Dropdown of data or input box.

Parameters

- **main** – Show form in main.
- **label** – Label identifier.
- **value** – Input value.
- **association** – Data association type from ‘Vertex’ or ‘Cell’.
- **data_type** – Type of data selectable from ‘Float’, ‘Integer’ or ‘Reference’.
- **data_group_type** – [Optional] Select from property_groups of type. ‘3D vector’, ‘Dip direction & dip’, ‘Strike & dip’, or ‘Multi-element’.
- **parent** – Parameter name corresponding to the parent object.
- **is_value** – Display the input box or dropdown menu.
- **prop** – Data entity selected in the dropdown menu if ‘is_value=False’.
- **optional** – Make optional if not None. Initial state provided by not None value. Can be either ‘enabled’ or ‘disabled’.

Returns

Ui_json compliant dictionary.

`geoh5py.ui_json.templates.file_parameter`(*main: bool = True, label: str = 'File choices', file_description: tuple = (), file_type: tuple = (), value: str = "", optional: str | None = None*) → dict

File loader for specific extensions.

Parameters

- **main** – Show form in main.
- **label** – Label identifier.
- **value** – Input value.
- **file_description** – Title used to describe each type.
- **file_type** – Extension of files to display.
- **optional** – Make optional if not None. Initial state provided by not None value. Can be either ‘enabled’ or ‘disabled’.

Returns

Ui_json compliant dictionary.

`geoh5py.ui_json.templates.float_parameter`(*main: bool = True, label: str = 'Float data', value: float = 1.0, vmin: float = 0.0, vmax: float = 100.0, precision: int = 2, line_edit: bool = True, optional: str | None = None*) → dict

Input box for float value.

Parameters

- **main** – Show form in main.
- **label** – Label identifier.
- **value** – Input value.
- **vmin** – Minimum value allowed.
- **vmax** – Maximum value allowed.
- **line_edit** – Allow line edit or spin box
- **optional** – Make optional if not None. Initial state provided by not None value. Can be either 'enabled' or 'disabled'.

Returns

Ui_json compliant dictionary.

`geoh5py.ui_json.templates.integer_parameter(main: bool = True, label: str = 'Integer data', value: int = 1, vmin: int = 0, vmax: int = 100, optional: str | None = None) → dict`

Input box for integer value.

Parameters

- **main** – Show ui in main.
- **label** – Label identifier.
- **value** – Input value.
- **vmin** – Minimum value allowed.
- **vmax** – Maximum value allowed.
- **optional** – Make optional if not None. Initial state provided by not None value. Can be either 'enabled' or 'disabled'.

Returns

Ui_json compliant dictionary.

`geoh5py.ui_json.templates.object_parameter`(*main: bool = True, label: str = 'Object', mesh_type: tuple = (UUID('4b99204c-d133-4579-a916-a9c8b98cfccb'), UUID('19730589-fd28-4649-9de0-ad47249d9aba'), UUID('58c4849f-41e2-4e09-b69b-01cf4286cded'), UUID('b020a277-90e2-4cd7-84d6-612ee3f25051'), UUID('9b08bb5a-300c-48fe-9007-d206f971ea92'), UUID('6a057fdc-b355-11e3-95be-fd84a7ffcb88'), UUID('c94968ea-cf7d-11eb-b8bc-0242ac130003'), UUID('7caebf0e-d16e-11e3-bc69-e4632694aa37'), UUID('77ac043c-fe8d-4d14-8167-75e300fb835a'), UUID('48f5054a-1c5c-4ca4-9048-80f36dc60a06'), UUID('deebe11a-b57b-4a03-99d6-8f27b25eb2a8'), UUID('17dbbfbb-3ee4-461c-9f1d-1755144aac90'), UUID('6832acf3-78aa-44d3-8506-9574a3510c44'), UUID('e79f449d-74e3-4598-9c9c-351a28b8b69e'), UUID('b99bd6e5-4fe1-45a5-bd2f-75fc31f91b38'), UUID('88087fb8-76ae-445b-9cdf-68dbce530404'), UUID('849d2f3e-a46e-11e3-b401-2776bdf4f982'), UUID('4ea87376-3ece-438b-bf12-3479733ded46'), UUID('202c5db1-a56d-4004-9cad-baafd8899406'), UUID('275ecee9-9c24-4378-bf94-65f3c5f9e163'), UUID('f26feba3-aded-494b-b9e9-b2bbcb298e1'), UUID('f495cd13-f09b-4a97-9212-2ea392aeb375'), UUID('0b639533-f35b-44d8-92a8-f70ecff3fd26'))*, *value: str | None = None, optional: str | None = None*) → dict

Dropdown menu of objects of specific types.

Parameters

- **main** – Show form in main.
- **label** – Label identifier.
- **value** – Input value.
- **mesh_type** – Type of selectable objects.
- **optional** – Make optional if not None. Initial state provided by not None value. Can be either 'enabled' or 'disabled'.

Returns

Ui_json compliant dictionary.

`geoh5py.ui_json.templates.optional_parameter`(*state: str*) → dict[str, bool]

Returns dictionary to make existing ui optional via .update() method.

Parameters

state – Initial state of optional parameter. Can be 'enabled' or 'disabled'.

`geoh5py.ui_json.templates.string_parameter`(*main: bool = True, label: str = 'String data', value: str = 'data', optional: str | None = None*) → dict

Input box for string value.

Parameters

- **main** – Show form in main.
- **label** – Label identifier.
- **value** – Input string value.

- **optional** – Make optional if not None. Initial state provided by not None value. Can be either 'enabled' or 'disabled'.

Returns

Ui_json compliant dictionary.

geoh5py.ui_json.utils

geoh5py.ui_json.utils.**collect**(*ui_json: dict[str, dict], member: str, value: Any = None*) → dict[str, Any]

Collects ui parameters with common field and optional value.

geoh5py.ui_json.utils.**container_group2name**(*value*)

geoh5py.ui_json.utils.**dependency_requires_value**(*ui_json: dict[str, dict], parameter: str*) → bool

Handles dependency and optional requirements.

If dependency doesn't require a value then the function returns False. But if the dependency does require a value, the return value is either True, or will take on the enabled state if the dependent parameter is optional.

Parameters

- **ui_json** – UI.json dictionary
- **parameter** – Name of parameter to check type.

geoh5py.ui_json.utils.**find_all**(*ui_json: dict[str, dict], member: str, value: Any = None*) → list[str]

Returns names of all collected parameters.

geoh5py.ui_json.utils.**flatten**(*ui_json: dict[str, dict]*) → dict[str, Any]

Flattens ui.json format to simple key/value pair.

geoh5py.ui_json.utils.**group_enabled**(*group: dict[str, dict]*) → bool

Return true if groupOptional and enabled are both true.

Parameters

group – UI.json dictionary

geoh5py.ui_json.utils.**group_optional**(*ui_json: dict[str, dict], group_name: str*) → bool

Returns groupOptional bool for group name.

geoh5py.ui_json.utils.**group_requires_value**(*ui_json: dict[str, dict], parameter: str*) → bool

True is groupOptional and group is enabled else False

Parameters

- **ui_json** – UI.json dictionary
- **parameter** – Name of parameter to check type.

geoh5py.ui_json.utils.**inf2str**(*value*)

geoh5py.ui_json.utils.**is_form**(*var*) → bool

Return true if dictionary 'var' contains both 'label' and 'value' members.

geoh5py.ui_json.utils.**is_uijson**(*ui_json: dict[str, dict]*)

Returns True if dictionary contains all the required parameters.

geoh5py.ui_json.utils.**list2str**(*value*)

`geoh5py.ui_json.utils.monitored_directory_copy(directory: str, entity: ObjectBase, copy_children: bool = True)`

Create a temporary geoh5 file in the monitoring folder and export entity for update.

Parameters

- **directory** – Monitoring directory
- **entity** – Entity to be updated
- **copy_children** – Option to copy children entities.

`geoh5py.ui_json.utils.none2str(value)`

`geoh5py.ui_json.utils.optional_requires_value(ui_json: dict[str, dict], parameter: str) → bool`

True if enabled else False.

Parameters

- **ui_json** – UI.json dictionary
- **parameter** – Name of parameter to check type.

`geoh5py.ui_json.utils.path2workspace(value)`

`geoh5py.ui_json.utils.requires_value(ui_json: dict[str, dict], parameter: str) → bool`

Check if a ui.json parameter requires a value (is not optional).

The required status of a parameter depends on a hierarchy of ui switches. At the top is the groupOptional switch, below that is the dependency switch, and on the bottom is the optional switch. When group optional is disabled all parameters in the group are not required, When the groupOptional is enabled the required status of a parameter depends first any dependencies and lastly on it's optional status.

Parameters

- **ui_json** – UI.json dictionary
- **parameter** – Name of parameter to check type.

`geoh5py.ui_json.utils.set_enabled(ui_json: dict, parameter: str, value: bool)`

Set enabled status for an optional or groupOptional parameter.

Parameters

- **ui_json** – UI.json dictionary
- **parameter** – Parameter name.
- **value** – Boolean value set to parameter's enabled member.

`geoh5py.ui_json.utils.str2inf(value)`

`geoh5py.ui_json.utils.str2list(value)`

`geoh5py.ui_json.utils.str2none(value)`

`geoh5py.ui_json.utils.truth(ui_json: dict[str, dict], name: str, member: str) → bool`

Return parameter's 'member' value with default value for non-existent members.

`geoh5py.ui_json.utils.workspace2path(value)`

geoh5py.ui_json.validation

```
class geoh5py.ui_json.validation.InputValidation(validators: dict[str, BaseValidator] | None = None,
                                                validations: dict[str, Any] | None = None,
                                                workspace: Workspace | None = None, ui_json:
                                                dict[str, Any] | None = None, validation_options:
                                                dict[str, Any] | None = None)
```

Bases: object

Validations on dictionary of parameters.

Attributes

validations

[Validations dictionary with matching set of input parameter keys.]

workspace (optional)

[Workspace instance needed to validate uuid types.]

ignore_requirements (optional): Omit raising error on 'required' validator.

Methods

validate_data(data)	Validates data of params and contents/type/shape/keys/reqs of values.
----------------------------	---

```
static infer_validations(ui_json: dict[str, Any] | None, validations: dict[str, dict] | None = None) →
dict
```

Infer necessary validations from ui json structure.

```
validate(name: str, value: Any, validations: dict[str, Any] | None = None)
```

Run validations on a given key and value.

Parameters

- **name** – Parameter identifier.
- **value** – Input parameter value.
- **validations** – [Optional] Validations provided on runtime

```
validate_data(data: dict[str, Any]) → None
```

Calls validate method on individual key/value pairs in input.

Parameters

data – Input data with known validations.

property validations

property validators

property workspace

2.3.7 geoh5py.workspace

geoh5py.workspace.workspace

```
class geoh5py.workspace.workspace.Workspace(h5file: str | Path | io.BytesIO = 'Analyst.geoh5', mode='a',
                                             **kwargs)
```

Bases: `AbstractContextManager`

The Workspace class manages all Entities created or imported from the *geoh5* structure.

The basic requirements needed to create a Workspace are:

Parameters

geoh5 – File name of the target *geoh5* file. A new project is created if the target file cannot be found on disk.

activate()

Makes this workspace the active one.

In case the workspace gets deleted, `Workspace.active()` safely returns `None`.

static active() → *Workspace*

Get the active workspace.

property attribute_map: dict

Mapping between names used in the *geoh5* database.

close()

Close the file and clear properties for future open.

property contributors: ndarray

`numpy.array` of `str` List of contributors name.

classmethod copy_property_groups(entity: *ObjectBase*, property_groups: list[*PropertyGroup*], data_map: dict)

copy_to_parent(entity, parent, copy_children: bool = True, omit_list: tuple = (), extent: np.ndarray | None = None, **kwargs)

Copy an entity to a different parent with copies of children.

Parameters

- **entity** – Entity to be copied.
- **parent** – Target parent to copy the entity under.
- **copy_children** – Copy all children of the entity.
- **omit_list** – List of property names to omit on copy
- **extent** – Clip object's copy by extent defined by a South-West and North-East corners.
- **kwargs** – Additional keyword arguments passed to the copy constructor.

Returns

The Entity registered to the workspace.

create_data(entity_class, entity_kwargs: dict, entity_type_kwargs: dict | *DataType*) → *Data* | None

Create a new *Data* entity with attributes.

Parameters

- **entity_class** – *Data* class.
- **entity_kwargs** – Properties of the entity.
- **entity_type_kwargs** – Properties of the entity_type.

Returns

The newly created entity.

create_entity(*entity_class*, *save_on_creation*: *bool* = *True*, ***kwargs*) → *Entity* | *None*

Function to create and register a new entity and its entity_type.

Parameters

- **entity_class** – Type of entity to be created
- **save_on_creation** – Save the entity to geoh5 immediately

Return entity

Newly created entity registered to the workspace

create_from_concatenation(*attributes*)

create_object_or_group(*entity_class*, *entity_kwargs*: *dict*, *entity_type_kwargs*: *dict*) → *Group* | *ObjectBase* | *None*

Create an object or a group with attributes.

Parameters

- **entity_class** – *ObjectBase* or *Group* class.
- **entity_kwargs** – Attributes of the entity.
- **entity_type_kwargs** – Attributes of the entity_type.

Returns

A new Object or Group.

property data: *list*[*data.Data*]

Get all active Data entities registered in the workspace.

deactivate()

Deactivate this workspace if it was the active one, else does nothing.

property distance_unit: *str*

str Distance unit used in the project.

fetch_array_attribute(*entity*: *Entity*, *key*: *str* = *'cells'*) → *ndarray*

Fetch attribute stored as structured array from the source geoh5.

Parameters

- **entity** – Unique identifier of target entity.
- **file** – *h5py.File* or name of the target geoh5 file
- **key** – Field array name

Returns

Structured array.

fetch_children(*entity*: *Entity* | *None*, *recursively*: *bool* = *False*) → *list*

Recover and register children entities from the geoh5.

Parameters

- **entity** – Parental entity.
- **recursively** – Recover all children down the project tree.
- **file** – h5py.File or name of the target geoh5 file.

Return list

List of children entities.

fetch_concatenated_attributes(*entity*: Concatenator | ConcatenatedObject) → dict | None

Fetch attributes of Concatenated entity.

Parameters

entity – Concatenator group or ConcatenateObject.

Returns

Dictionary of attributes.

fetch_concatenated_list(*entity*: Group | ObjectBase, *label*: str) → list | None

Fetch list of data or indices of ConcatenatedData entities.

Parameters

- **entity** – Concatenator group.
- **label** – Label name of the h5py.Group

Returns

List of concatenated Data names.

fetch_concatenated_values(*entity*: Group | ObjectBase, *label*: str) → tuple | None

Fetch data under the ConcatenatedData Data group of an entity.

Parameters

- **entity** – Concatenator group.
- **label** – Name of the target data.

Returns

Index array and data values for the target label.

fetch_file_object(*uid*: uuid.UUID, *file_name*: str) → bytes | None

Fetch an image from file name.

Parameters

uid – Unique identifier of target data object.

Returns

Array of values.

fetch_metadata(*uid*: uuid.UUID, *argument*='Metadata') → dict | None

Fetch the metadata of an entity from the source geoh5.

Parameters

- **uid** – Entity uid containing the metadata.
- **argument** – Optional argument for other json-like attributes.

Returns

Dictionary of values.

fetch_or_create_root()

fetch_property_groups(*entity*: [Entity](#)) → list[[PropertyGroup](#)]

Fetch all property_groups on an object from the source geoh5

Parameters

entity – Target object

Returns

List of PropertyGroups

fetch_type(*uid*: *UUID*, *entity_type*: *str*) → dict

Fetch attributes of a specific entity type. :param uid: Unique identifier of the entity type. :param entity_type: One of ‘Data’, ‘Object’ or ‘Group’

fetch_values(*entity*: [Entity](#)) → np.ndarray | str | float | None

Fetch the data values from the source geoh5.

Parameters

entity – Entity with ‘values’.

Returns

Array of values.

finalize() → None

Deprecate method finalize

Parameters

file – *h5py.File* or name of the target geoh5 file

find_data(*data_uid*: *uuid.UUID*) → [Entity](#) | None

Find an existing and active Data entity.

find_entity(*entity_uid*: *uuid.UUID*) → [Entity](#) | None

Get all active entities registered in the workspace.

find_group(*group_uid*: *uuid.UUID*) → [group.Group](#) | None

Find an existing and active Group object.

find_object(*object_uid*: *uuid.UUID*) → [object_base.ObjectBase](#) | None

Find an existing and active Object.

find_type(*type_uid*: *uuid.UUID*, *type_class*: *type[EntityType]*) → [EntityType](#) | None

Find an existing and active EntityType

Parameters

type_uid – Unique identifier of target type

property ga_version: *str*

str Version of Geoscience Analyst software.

property geoh5: *File*

Instance of *h5py.File*.

get_entity(*name*: *str* | *uuid.UUID*) → list[[Entity](#) | None]

Retrieve an entity from one of its identifier, either by name or *uuid.UUID*.

Parameters

name – Object identifier, either name or *uuid*.

Returns

List of entities with the same given name.

property groups: `list[groups.Group]`

Get all active Group entities registered in the workspace.

property h5file: `str | Path | io.BytesIO`

Str

Target *geoh5* file name with path.

property list_data_name: `dict[uuid.UUID, str]`

dict of `uuid.UUID` keys and name values for all registered Data.

property list_entities_name: `dict[uuid.UUID, str]`

Returns

dict of `uuid.UUID` keys and name values for all registered Entities.

property list_groups_name: `dict[uuid.UUID, str]`

dict of `uuid.UUID` keys and name values for all registered Groups.

property list_objects_name: `dict[uuid.UUID, str]`

dict of `uuid.UUID` keys and name values for all registered Objects.

load_entity(*uid*: `uuid.UUID`, *entity_type*: `str`, *parent*: `Entity | None = None`) → `Entity | None`

Recover an entity from geoh5.

Parameters

- **uid** – Unique identifier of entity
- **entity_type** – One of entity type ‘group’, ‘object’, ‘data’ or ‘root’

Return entity

Entity loaded from geoh5

property name: `str`

str Name of the project.

property objects: `list[objects.ObjectBase]`

Get all active Object entities registered in the workspace.

open(*mode*: `str | None = None`) → `Workspace`

Open a geoh5 file and load the tree structure.

Parameters

mode – Optional mode of `h5py.File`. Defaults to ‘r+’.

Returns

self

remove_children(*parent*, *children*: `list`)

Remove a list of entities from a parent. The target entities remain present on file.

remove_entity(*entity*: `Entity`)

Function to remove an entity and its children from the workspace.

remove_none_referents(*referents*: `dict[uuid.UUID, ReferenceType]`, *rtype*: `str`)

Search and remove deleted entities

remove_recursively(*entity*: `Entity`)

Delete an entity and its children from the workspace and geoh5 recursively

property repack: bool

Flag to repack the file after data deletion

property root: [Entity](#) | None

[RootGroup](#) entity.

save_entity(entity: [Entity](#), add_children: bool = True) → None

Save or update an entity to geoh5.

Parameters

- **entity** – Entity to be written to geoh5.
- **add_children** – Add children entities to geoh5.
- **file** – `h5py.File` or name of the target geoh5

save_entity_type(entity_type: [EntityType](#)) → None

Save or update an entity_type to geoh5.

Parameters

entity_type – Entity to be written to geoh5.

static str_from_type(entity) → str | None

property types: list[[EntityType](#)]

Get all active entity types registered in the workspace.

update_attribute(entity: [Entity](#) | [EntityType](#), attribute: str, channel: str | None = None, **kwargs) → None

Save or update an entity to geoh5.

Parameters

- **entity** – Entity to be written to geoh5.
- **attribute** – Name of the attribute to get updated to geoh5.
- **channel** – Optional channel argument for concatenated data and index.

property version: float

float Version of the geoh5 file format.

property workspace: [Workspace](#)

This workspace instance itself.

`geoh5py.workspace.workspace.active_workspace(workspace: Workspace)`

2.4 GEOH5 Format

2.4.1 About

The GEOH5 format aims to

provide a robust means of handling large quantities of diverse data required in geoscience. The file structure builds on the generic qualities of the

Geoscience ANALYST data model, and attempts to maintain a certain level of simplicity and consistency throughout. It is based entirely on free and open [HDF5 technology](#). Given that this specification is public, the file format could, with further investment and involvement, become a useful exchange format for the broader geoscientific community.

Why GEOH5?

- Leverages properties
 - Fast I/O, compression, cross-platform
- Content readable and writable
 - We recommend

mend
us-
ing
HD-
FView,
along
with
Geo-
science
AN-
A-
LYST,
when
learn-
ing
the
for-
mat.

• **Easily extensible to**

It
is
in-
tended
for
Geo-
science
AN-
A-
LYST
to
pre-
serve
data
it
does
not
un-
der-
stand
(and
gen-
er-
ally
be
very
tol-
er-
ant
with
re-
gards
to
miss-

ing
in-
for-
ma-
tion)
when

loading and saving geoh5 files. This will allow third parties to write to this format fairly easily, as well as include additional information not included in this spec for their own purposes. In the current implementation, Geoscience ANALYST automatically removes unnecessary information on save.

2.4.2 Defin

The
fol-
low-
ing
sec-
tions
de-
fine
the
struc-
ture
and
com-
po-
nents
mak-
ing
up
the
GEOH5
file
for-
mat.

Workspace

The
bulk
of
the
data
is
ac-
ces-
si-
ble
both
di-
rectly
by

UUID
through
the
flat
HDF5
groups
or
through
a
**hi-
er-
ar-
chy
of
hard
links**
struc-
tured
as
fol-
low:

Data

Flat
con-
tainer
for
all
data
en-
ti-
ties

Groups

Flat
con-
tainer
for
all
group
en-
ti-
ties

Objects

Flat
con-
tainer
for
all
ob-
ject
en-
ti-
ties

Root
Optional
hard
link
to
workspace
group,
top
of
group
hi-
er-
ar-
chy.

Types

- *Data Types:*
Flat container for all data types
- *Group Types:*
Flat container for all group types
- *Object Types:*
Flat container for all object types

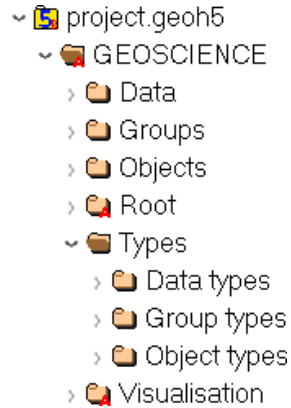


Fig. 2.1: As seen in HD-FView

While all groups, objects and data entities are written into their respective base folder, they also hold links to their children entities to allow for traversals. There is no data duplication, merely multiple references (pointers) to the data storage on file. Types are shared (and thus generally written to file first). All groups, objects and data must include a hard link to their type.

Attributes

Version

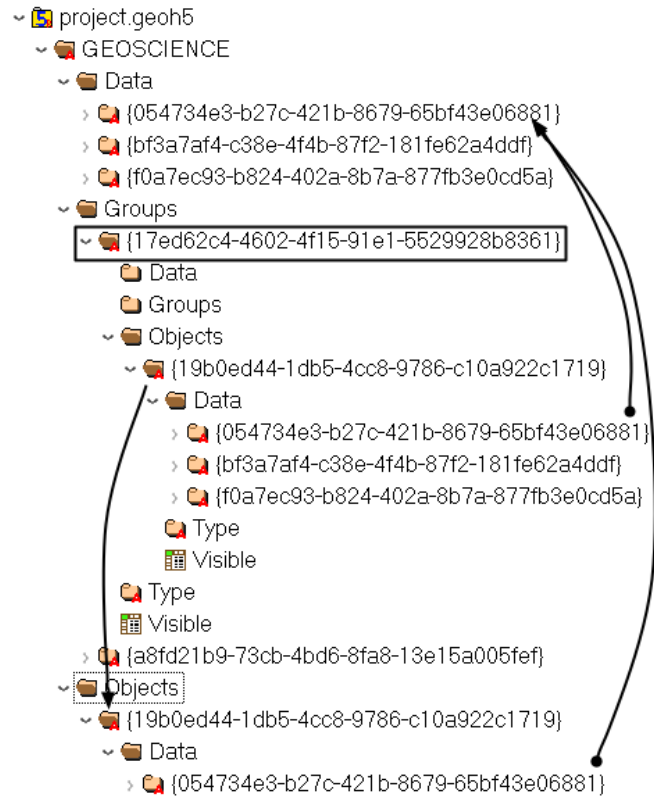
double Version of specification used by this file

Distance unit

str *feet* or (default) *metres* Distance unit of all data enclosed

Contributors

1D array of str (Optional) List of users who contributed to this workspace



Groups

Groups are simple container for other groups and objects. They are often used to assign special meanings to a collection of entities or to create specialized software functionality.

Attributes

Name

str Name of the object displayed in the project tree.

ID

str, *UUID* Unique identifier of the group.

Visible

int, 0 or (default) 1 Set visible in the 3D camera (checked in the object tree).

Public

int, 0 or (default) 1 Set accessible in the object tree and other parts of the the user interface.

Clipping IDs

1D array of *UUID* (Optional) List of unique identifiers of clipping plane objects.

Allow delete

int, 0 or (default) 1 (Optional) User interface allows deletion.

Allow move

int, 0 or (default) 1 (Optional) User interface allows moving to another parent group.

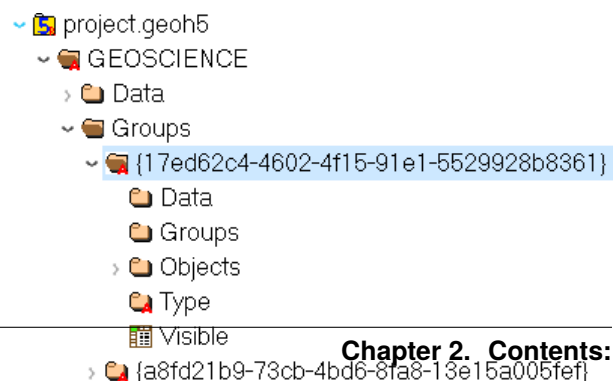
Allow rename

int, 0 or (default) 1 (Optional) User interface allows renaming.

Metadata

(int, optional) (Optional) Any additional text attached to the group.

The following section describes the supported group types.



Group Types

To be further documented

Container

UUID : {61FBB4E8-A480-11E3-8D5A-2776BDF4F982}

Simple container with no special meaning. Default in Geoscience ANALYST.

Drillholes

UUID : {825424FB-C2C6-4FEA-9F2B-6CD00023D393}

Container restricted to containing drillhole objects, and which may provide convenience functionality for the drillholes within.

No Type (Root)

UUID : {dd99b610-be92-48c0-873c-5b5946ea2840}

The Root group defines the tree structure used in Geoscience ANALYST describing the parent-child relationships of entities. If absent, any Groups/Objects/Data will be brought into Geoscience ANALYST under the workspace group, still respecting any defined hierarchy links.

SimPEG

UUID : {55ed3daf-c192-4d4b-a439-60fa987fe2b8}

Container group for SimPEG inversions. Contains

Datasets

Metadata

json formatted string

Dictionary of inversion options.

options

ui.json formatted string

Dictionary holding the corresponding ui.json.

Tools

UUID : {a2befc38-3207-46aa-95a2-16b40117a5d8}

Group for slicer and label objects.

Not yet geoh5py implemented

To be further documented

Maxwell

UUID : {1c4122b2-8e7a-4ec3-8d6e-c818495adac7}

Group for Maxwell plate modeling application.

Not yet geoh5py implemented

To be further documented

GIFTools

GIFtools group containers

GIFtools Project

UUID : {585b3218-c24b-41fe-ad1f-24d5e6e8348a}

Not yet geoh5py implemented

To be documented

GIF Executables

UUID : {afae95ef-c2a7-4aec-9800-0d19bd2c2c07}

Not yet geoh5py implemented

To be documented

gzinv3d

UUID : {20eb4ff8-bdfe-43f3-8745-f418dcc9e14a}

Not yet geoh5py implemented

To be documented

gzfor3d

UUID : {a4857df0-d175-4824-ac5d-cecfdcc2f20b}

Not yet geoh5py implemented

To be documented

magfor3d

UUID : {6b8189ac-a479-4fe7-b4fc-92279aee5a41}

Not yet geoh5py implemented

To be documented

maginv3d

UUID : {b99e8db8-e118-4042-864e-9e1128f2d1e6}

Not yet geoh5py implemented

To be documented

mvifwd

UUID : {14c41f47-bcee-4a63-8192-fa42a1741052}

Not yet geoh5py implemented

To be documented

ggfor3d

UUID : {c8a8424d-ab12-482e-82ee-b198fcfd5859}

Not yet geoh5py implemented

To be documented

gginv3d

UUID : {0f080369-b3a3-464c-83fa-9b3c1efa9895}

Not yet geoh5py implemented

To be documented

mviinv

UUID : {9472b5cb-a285-4257-a2e8-68a3d33aa1f2}

Not yet geoh5py implemented

To be documented

octgrvde

UUID : {4e043415-a0ea-4cef-bf89-2771e27b346c}

Not yet geoh5py implemented

To be documented

octmagde

UUID : {f8217512-296d-4cc0-afcb-6c07a20581fe}

Not yet geoh5py implemented

To be documented

dcinv3d

UUID : {ae416ab8-0e72-4f37-8873-5cc0909433bb}

Not yet geoh5py implemented

To be documented

ipinv3d

UUID : {9f9543a0-e857-4a56-ab66-9f21e2b002c6}

Not yet geoh5py implemented

To be documented

e3dmt

UUID : {8cf239e3-63a6-4813-adf8-9714293b602e}

Not yet geoh5py implemented

To be documented

dcoc tree_inv

UUID : {54d296de-0588-472c-9a62-480098303394}

Not yet geoh5py implemented

To be documented

dcoc tree_fwd

UUID : {A522D641-6CB7-421B-836B-A14C0D9C7801}

Not yet geoh5py implemented

To be documented

ipoc tree_inv

UUID : {d9fd455e-ea94-40f5-9d86-e7c49c7b5005}

Not yet geoh5py implemented

To be documented

dcipf3d

UUID : {59b5338d-596c-4049-9aa4-6979700e00ff}

Not yet geoh5py implemented

To be documented

Geoscience INTEGRATOR Groups

Geoscience INTEGRATOR

UUID : {61449477-3833-11e4-a7fb-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Project

UUID : {56f6f03e-3833-11e4-a7fb-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Query Group

UUID : {85756113-592a-4088-b374-f32c8fac37a2}

Not yet geoh5py implemented

To be documented

Neighbourhoods

UUID : {2a5b7faa-41d1-4437-afac-934933eae6eb}

Not yet geoh5py implemented

To be documented

Map File Group

UUID : {1f684938-2baf-4a01-ac71-e50c30cc0685}

Not yet geoh5py implemented

To be documented

Maps Group

UUID : {4d65f8c3-a015-4c01-b411-412c0f4f0884}

Not yet geoh5py implemented

To be documented

Airborne

UUID : {812f3b2a-fdae-4752-8391-3b657953a983}

Not yet geoh5py implemented

To be documented

Ground

UUID : {a9d05630-7a80-4bda-89a2-feca0dc7a83e}

Not yet geoh5py implemented

To be documented

Borehole

UUID : {f6f011a9-2e52-4f99-b842-a524ad9fdf03}

Not yet geoh5py implemented

To be documented

Geoscience INTEGRATOR Themes

Data

UUID : {51fdf764-3833-11e4-a7fb-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Interpretation

UUID : {05e96011-3833-11e4-a7fb-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Microseismic

UUID : {2bddbaaf-3829-11e4-8654-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Ground Deformation

UUID : {7ade974c-3829-11e4-9cce-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Production Area

UUID : {55ccb6d9-016c-47cd-824f-077214dc44db}

Not yet geoh5py implemented

To be documented

Blasting

UUID : {e2040afa-3829-11e4-a70e-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Stress

UUID : {460e31c8-3829-11e4-a70e-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Drillholes and Wells

UUID : {5d9b6a8c-3829-11e4-93fc-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Mobile Equipment

UUID : {e7f63d21-3833-11e4-a7fb-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Fixed Plant

UUID : {fad14ac4-3833-11e4-a7fb-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Earth Model Points

UUID : {fcd708da-3833-11e4-a7fb-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Earth Model Regular 3D Grids

UUID : {79b41607-ffa2-4825-a270-44dd48807a03}

Not yet geoh5py implemented

To be documented

Observation Points

UUID : {f65e521c-a763-427b-97bf-d0b4e5689e0d}

Not yet geoh5py implemented

To be documented

Targets & Anomalies

UUID : {e41c2308-0f35-47dd-8562-d0fd354406f8}

Not yet geoh5py implemented

To be documented

Targets

UUID : {af0925ba-3dc5-4fe6-ab35-9e0ef568023f}

Not yet geoh5py implemented

To be documented

Anomalies

UUID : {51bcc3e9-1d66-4c83-847e-5c852fc9de58}

Not yet geoh5py implemented

To be documented

Fusion Model

UUID : {3d69be5b-3833-11e4-a7fb-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Deformation

UUID : {5caf35fa-3d0e-11e4-939f-f5f83219c4e0}

Not yet geoh5py implemented

To be documented

Mine Production

UUID : {7508bc11-3829-11e4-9cce-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Earth Models

UUID : {adee3b2a-3829-11e4-a70e-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Mine Models

UUID : {e53a8b3e-3829-11e4-a70e-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Samples

UUID : {1cde9996-cda7-40f0-8c20-faeb4e926748}

Not yet geoh5py implemented

To be documented

Geochemistry & Mineralogy

UUID : {ed00094f-3da1-485f-8c4e-b52f6f171ea4}

Not yet geoh5py implemented

To be documented

Rock Properties

UUID : {cbeb3920-a1a9-46f8-ab2b-7dfd79c8a00}

Not yet geoh5py implemented

To be documented

Incidents

UUID : {136cb431-c7d2-4992-a5ab-46a6e16b6726}

Not yet geoh5py implemented

To be documented

Mine Infrastructure

UUID : {cff33bb0-ef43-4b06-8070-266940ab9d06}

Not yet geoh5py implemented

To be documented

3D Structural Surfaces

UUID : {a246f9e0-2b67-4efd-bd3d-742bfe06178b}

Not yet geoh5py implemented

To be documented

3D Domains

UUID : {f69979b0-5ba1-417a-93d4-778146049014}

Not yet geoh5py implemented

To be documented

3D Geological Contact Surfaces

UUID : {0bf96ee1-7fa4-41a2-bc8a-7cd76426ba18}

Not yet geoh5py implemented

To be documented

Remote Sensing and Air Photos

UUID : {386f2c57-1893-40bb-bd1c-95552b90e514}

Not yet geoh5py implemented

To be documented

Inversions

UUID : {7a7b14af-23d9-4897-9cdb-8d586fefa025}

Not yet geoh5py implemented

To be documented

Topography

UUID : {c162ddd2-a9de-4dac-b6a2-3cc6e011d7c3}

Not yet geoh5py implemented

To be documented

Culture

UUID : {dd51ca09-34d7-4c30-a0d0-ef9e61ea5e9d}

Not yet geoh5py implemented

To be documented

Claims, boundaries

UUID : {6e430b33-4ab8-45c1-896d-c47525185ce0}

Not yet geoh5py implemented

To be documented

Ventilation

UUID : {d049e5a0-aadb-4448-a0f1-fe560c6d26f9}

Not yet geoh5py implemented

To be documented

Gas Monitoring

UUID : {bc8540b0-d814-46ac-b897-b5a528d5d1d6}

Not yet geoh5py implemented

To be documented

Ventilation & Gas Monitoring

UUID : {8ebd9b52-801e-4461-b7e6-e1aa0a8742b3}

Not yet geoh5py implemented

To be documented

Other

UUID : {79b61598-7385-4b63-8513-636ecde9c150}

Not yet geoh5py implemented

To be documented

Airborne

UUID : {3d0e8578-7764-48cf-8db8-6c83d6411762}

Not yet geoh5py implemented

To be documented

Ground

UUID : {47d6f059-b56a-46c7-8fc7-a0ded87360c3}

Not yet geoh5py implemented

To be documented

Integrator Borehole

UUID : {9c69ef80-b45c-4f5c-ac55-996a99dc299f}

Not yet geoh5py implemented

To be documented

Geophysics

UUID : {151778d9-6cc0-4e72-ba08-2a80a4fb967f}

Not yet geoh5py implemented

To be documented

Geotechnical

UUID : {391a616b-3833-11e4-a7fb-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Equipment

UUID : {8beac9ff-3829-11e4-8654-fcddabfddab1}

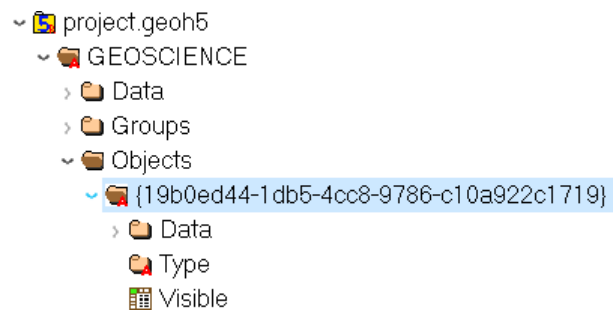
Not yet geoh5py implemented

To be documented

Note: Though this file format technically allows objects/groups to appear within multiple groups simultaneously (overlapping lists), this is not currently supported by Geoscience ANALYST.

Objects

Objects are containers for Data with spatial information. Most (not all) object geometry is described in terms of vertices (3D locations) and cells (groupings of vertices such as triangles or segments). The exact requirements and interpretation depends on the type.



Attributes

Name

str Name of the object displayed in the project tree.

ID

str Unique identifier (*UUID*) of the group.

Visible

int, 0 or (default) 1 Set visible in the 3D camera (checked in the object tree).

Public

int, 0 or (default) 1 Set accessible in the object tree and other parts of the the user interface.

Clipping IDs

1D array of UUID (Optional) List of unique identifiers of clipping plane objects.

Allow delete

int, 0 or (default) 1 (Optional) User interface allows deletion.

Allow move

int, 0 or (default) 1 (Optional) User interface allows moving to another parent group.

Allow rename

int, 0 or (default) 1 (Optional) User interface allows renaming.

Metadata

(int, optional) (Optional) Any additional text attached to the group.

The following section describes the supported object types and their specific attributes.

ANALYST Objects

Entities with spatial information used to store data.

Points

UUID : {202C5DB1-A56D-4004-9CAD-BAAFD8899406}

3-D scatter points object defined by vertices with fixed coordinates in Cartesian system (x, y and z).

Datasets

Vertices

1D composite array

[x double, y double, z double]

Curve

UUID : {6A057FDC-B355-11E3-95BE-FD84A7FFCB88}

Polyline object defined by a series of line segments (cells) connecting vertices. Data can be associated to either the vertices or cells.

Attributes

Current line property ID

str, *UUID*

Unique identifier of a reference data for naming of curve parts.

Datasets

Cells

Array of `int32`, shape(N, 2)

Array defining the connection (line segment) between pairs of vertices.

Surface

UUID : {F26FEBA3-ADED-494B-B9E9-B2BBCBE298E1}

Triangulated mesh object defined by cells (triangles) and vertices.

Datasets

Cells

Array of `int32`, shape(N, 3)

Array defining the connection between triplets of vertices, representing triangles.

Block model

UUID : {B020A277-90E2-4CD7-84D6-612EE3F25051}

Rectilinear grid of cells defined along three orthogonal axes (U,V and Z) of length `nU`, `nV` and `nZ` respectively. The conversion between the array coordinates of a cell to a 1-D vector index can be calculated from

```
cell index = k + i*nZ + j*nU*nZ
```

Without rotation angles, U points eastwards, V points northwards, and Z points upwards. Since their geometry is defined entirely by the additional data described below, block models do not require a `Vertices` or `Cells` dataset.

Datasets

U cell delimiters

array of `double`, shape(`nU`,)

Distances of cell edges from origin along the U axis (first value should be 0)

V cell delimiters

array of `double`, shape(`nV`,)

Distances of cell edges from origin along the V axis (first value should be 0)

Z cell delimiters

array of double, shape(nZ,)

Distances of cell edges from origin upwards along the vertical axis (first value should be 0)

Attributes**Origin**

composite type

[X double, Y double, Z double]

Origin point of grid

Rotation

double (default) 0 Counterclockwise angle (degrees) of rotation around the vertical axis in degrees.

2D Grid**UUID : {48f5054a-1c5c-4ca4-9048-80f36dc60a06}**

Rectilinear grid of cells defined along two orthogonal axes (U and V) of length nU and nV. The conversion between the grid coordinates of a cell to its 1-D vector index can be calculated from

```
cell index = i + j*nU
```

Without rotation angles, U points eastwards and V points northwards. Since their geometry is defined entirely by the additional data described below, 2D grids do not require a Vertices or Cells dataset.

Attributes**Origin**

composite type

[X double, Y double, Z double]

Origin point of the grid.

U Size

double Length of U axis

U Count

double Number of cells along U axis

V Size

double Length of V axis

V Count

double Number of cells along V axis

Rotation

double (Optional) Counterclockwise angle (degrees) of rotation around the vertical axis at the Origin.

Vertical

char, 0(false, default) or 1(true)) (Optional) If true, V axis is vertical (and rotation defined around the V axis)

Drillhole

UUID : {7CAEBF0E-D16E-11E3-BC69-E4632694AA37}

Object representing boreholes defined by a collar location and survey parameters. Vertices represent points along the drillhole path (support for data rather than the drillhole geometry itself) and must have a `Depth` property value. Cells contain two vertices and represent intervals along the drillhole path (and are a support for interval data as well). Cells may overlap with each other to accommodate the different sampling intervals of various data.

Attributes

Collar

composite type, shape(3,)

[*X* double, *Y* double, *Z* double]

Collar location

Datasets

Surveys

composite array, shape(3,)

[*Depth* double, *Dip* double, *Azimuth* double]

Survey locations

Trace

1D composite array

[*X* double, *Y* double, *Z* double]

Points forming the drillhole path from collar to end of hole. Must contain at least two points.

Geoimage

UUID : {77AC043C-FE8D-4D14-8167-75E300FB835A}

Not yet geoh5py implemented

To be further documented

Vertices represent the four corners of the geolocated image. No cell data. An object-associated file-type data containing the image to display is expected to exist under this object.

Note: Should be arranged as a rectangle currently, since Geoscience ANALYST does not currently support skewed images.

Label

UUID : {E79F449D-74E3-4598-9C9C-351A28B8B69E}

Not yet geoh5py implemented

To be further documented

Has no vertices nor cell data

Attributes

Target position

composite type, shape(3,)

[X double, Y double, Z double]

The target location of the label

Label position

composite type, shape(3,)

[X double, Y double, Z double] (Optional - Defaults to same as target position) The location where the text of the label is displayed

Slicer

UUID : {238f961d-ae63-43de-ab64-e1a079271cf5}

Not yet geoh5py implemented

To be further documented

Target

UUID : {46991a5c-0d3f-4c71-8661-354558349282}

Not yet geoh5py implemented

To be further documented

ioGAS Points

UUID : {d133341e-a274-40e7-a8c1-8d32fb7f7eaf}

Not yet geoh5py implemented

To be further documented

Maxwell Plate

UUID : {878684e5-01bc-47f1-8c67-943b57d2e694}

Not yet geoh5py implemented

To be further documented

Octree

UUID : {4ea87376-3ece-438b-bf12-3479733ded46}

Semi-structured grid that stores cells in a tree structure with eight octants.

Datasets

Octree Cells

composite type, shape(N, 4)

[*I* integer, *J* integer, *K* integer, *NCells* integer]

Array defining the position (I, J, K) and size (NCells) of every cell within the base octree grid.

Attributes

NU

integer Number of base cells along the U-axis.

NV

integer Number of base cells along the V-axis.

NW

integer Number of base cells along the W-axis.

Origin

composite type, shape(3,)

[*X* double, *Y* double, *Z* double]

Origin point of the grid.

Rotation

double (default) 0 Counterclockwise angle (degrees) of rotation around the vertical axis in degrees.

U Cell Size

double Base cell dimension along the U-axis.

V Cell Size

double Base cell dimension along the V-axis.

W Cell Size

double Base cell dimension along the W-axis.

Text Object

UUID : {c00905d1-bc3b-4d12-9f93-07fcf1450270}

Not yet geoh5py implemented

To be further documented

Potential Electrode

UUID : {275ecee9-9c24-4378-bf94-65f3c5fbe163}

Curve object representing the receiver electrodes of a direct-current resistivity survey.

Datasets

Metadata

json formatted string

Dictionary defining the link between the source and receiver objects.

- “Current Electrodes” uuid: Identifier for the linked *Current Electrode*
- “Potential Electrodes” uuid: Identifier for the linked *Potential Electrode*

Requirements

A-B Cell ID

Data entity

Reference data named “A-B Cell ID” with association=CELL expected. The values define the source dipole (cell) from the *Current Electrode* to every potential measurement.

Current Electrode

UUID : {9b08bb5a-300c-48fe-9007-d206f971ea92}

Curve object representing the transmitter electrodes of a direct-current resistivity survey.

Datasets

Metadata

json formatted string

Dictionary defining the link between the source and receiver objects. Same definition as the *Potential Electrode* object.

Requirements

A-B Cell ID

Data entity

Reference data named “A-B Cell ID” with `association=CELL` defining a unique identifier to every unique dipole sources. For “pole” sources, the `cell` attribute references twice to the same vertex.

VP Model

UUID : {7d37f28f-f379-4006-984e-043db439ee95}

Not yet geoh5py implemented

To be further documented

Airborne EM

UUID : {fdf7d01e-97ab-43f7-8f2c-b99cc10d8411}

Not yet geoh5py implemented

To be further documented

Airborne TEM Rx

UUID : {19730589-fd28-4649-9de0-ad47249d9aba}

Curve object representing an array of time-domain electromagnetic receiver dipoles.

Attributes

Target position

composite type

Datasets

Metadata

json formatted string

Dictionary of survey parameters shared with the *Transmitters*. The following items are core parameters stored under the “EM Dataset” key.

- **“Channels”: list of double**
Time channels at which data are recorder.
- **“Input type”: string**
Type of survey from “Rx”, “Tx” or “Tx and Rx”
- **“Loop radius”: double**
Transmitter loop radius.
- **“Property groups”: list of uuid**
Reference to property groups containing data at every channel.

- **“Receivers”: uuid**
Unique identifier referencing to itself.
- **“Survey type”: string**
Defaults to “Airborne TEM”.
- **“Transmitters”: uuid**
Unique identifier referencing to the linked transmitters entity.
- **“Unit”: string**
Sampling units, must be one of “Seconds (s)”, “Milliseconds (ms)”, “Microseconds (us)” or “Nanoseconds (ns)”.
- **“Crossline offset property” uuid OR “Crossline offset value” double:**
Offline offset between the receivers and transmitters, either defined locally on vertices as a property OR globally as a constant value.
- **“Inline offset property” uuid OR “Crossline offset value” double:**
Inline offset between the receivers and transmitters, either defined locally on vertices as a property OR globally as a constant value.
- **“Inline offset property” uuid OR “Crossline offset value” double:**
Vertical offset between the receivers and transmitters, either defined locally on vertices as a property OR globally as a constant value.
- **“Yaw property” uuid OR “Yaw value” double:**
Rotation (angle) of the transmitter loop as measured on the UV-plane (+ clockwise), either defined locally on vertices as a property OR globally as a constant value.
- **“Pitch property” uuid OR “Pitch value” double:**
Tilt angle of the transmitter loop as measured on the VW-plane (+ nose up), either defined locally on vertices as a property OR globally as a constant value.
- **“Roll property” uuid OR “Roll value” double:**
Banking angle of the transmitter loop as measured on the UW-plane (+ right-wing down), either defined locally on vertices as a property OR globally as a constant value.
- **“Waveform” dict:**
 - **“Discretization” array of double, shape(N, 2):**
Array of times and normalized currents (Amp) describing the source impulse over a discrete interval (e.g. [[t_1, c_1], [t_2, c_2], ..., [t_N, c_N]])
 - **“Timing mark” double:**
Reference timing mark measured from the beginning of the “Discretization”. Generally used as the reference (t_i=0.0) for the provided data channels: (-) on-time an (+) off-time.

Airborne TEM Tx

UUID : {58c4849f-41e2-4e09-b69b-01cf4286cded}

Curve object representing an array of time-domain electromagnetic transmitter loops.

Datasets

Metadata

json formatted string

See definition from the *Airborne TEM Rx* object. The “Transmitters” uuid value should point to itself, while the “Receivers” uuid refers the linked *Airborne TEM Rx* object.

Airborne FEM Rx

UUID : {b3a47539-0301-4b27-922e-1dde9d882c60}

Not yet geoh5py implemented

To be further documented

Airborne FEM Tx

UUID : {a006cf3e-e24a-4c02-b904-2e57b9b5916d}

Not yet geoh5py implemented

To be further documented

Airborne Gravity

UUID : {b54f6be6-0eb5-4a4e-887a-ba9d276f9a83}

Not yet geoh5py implemented

To be further documented

Airborne Magnetics

UUID : {4b99204c-d133-4579-a916-a9c8b98cfcdb}

Not yet geoh5py implemented

To be further documented

Ground Gravity

UUID : {5ffa3816-358d-4cdd-9b7d-e1f7f5543e05}

Not yet geoh5py implemented

To be further documented

Ground Magnetics

UUID : {028e4905-cc97-4dab-b1bf-d76f58b501b5}

Not yet geoh5py implemented

To be further documented

Ground Gradient IP

UUID : {68b16515-f424-47cd-bb1a-a277bf7a0a4d}

Not yet geoh5py implemented

To be further documented

Ground EM

UUID : {09f1212f-2bdd-4dea-8bbd-f66b1030dfcd}

Not yet geoh5py implemented

To be further documented

Ground TEM Rx

UUID : {41018a45-01a0-4c61-a7cb-9f32d8159df4}

Not yet geoh5py implemented

To be further documented

Ground TEM Tx

UUID : {98a96d44-6144-4adb-afbe-0d5e757c9dfc}

Not yet geoh5py implemented

To be further documented

Ground TEM Rx (large-loop)

UUID : {deebe11a-b57b-4a03-99d6-8f27b25eb2a8}

Not yet geoh5py implemented

To be further documented

Ground TEM Tx (large-loop)

UUID : {17dbbfb-3ee4-461c-9f1d-1755144aac90}

Not yet geoh5py implemented

To be further documented

Ground FEM Rx

UUID : {a81c6b0a-f290-4bc8-b72d-60e59964bfe8}

Not yet geoh5py implemented

To be further documented

Ground FEM Tx

UUID : {f59d5a1c-5e63-4297-b5bc-43898cb4f5f8}

Not yet geoh5py implemented

To be further documented

Magnetotellurics

UUID : {b99bd6e5-4fe1-45a5-bd2f-75fc31f91b38}

Points object representing a magnetotelluric survey.

Metadata

json formatted string

Dictionary of survey parameters. The following items are core parameters stored under the “EM Dataset” key.

- **“Channels”: list of double**
Frequency channels at which data are recorder.
- **“Input type”: string**
Static field set to “Rx only”
- **“Property groups”: list of uuid**
Reference to property groups containing data at every channel.
- **“Receivers”: uuid**
Reference to itself.
- **“Survey type”: string**
Static field set to “Magnetotellurics”
- **“Unit”: string**
Sampling units, must be one of “Hertz (Hz)”, “KiloHertz (kHz)”, “MegaHertz (MHz)” or “Gigahertz (GHz)”.

Tipper Rx

UUID : {0b639533-f35b-44d8-92a8-f70ecff3fd26}

Curve object representing a tipper survey.

Metadata

json formatted string

Dictionary of survey parameters. The following items are core parameters stored under the “EM Dataset” key.

- **“Channels”: list of double**
Frequency channels at which data are recorder.
- **“Input type”: string**
Static field set to “Rx and base stations”
- **“Property groups”: list of uuid**
Reference to property groups containing data at every channel.
- **“Receivers”: uuid**
Reference to itself.
- **“Base stations: uuid**
Reference to *Tipper Base stations*
- **“Survey type”: string**
Static field set to “Magnetotellurics”
- **“Unit”: string**
Sampling units, must be one of “Hertz (Hz)”, “KiloHertz (kHz)”, “MegaHertz (MHz)” or “Gigahertz (GHz)”.

Tipper Base stations

UUID : {f495cd13-f09b-4a97-9212-2ea392aeb375}

Points object representing a tipper survey.

Metadata

json formatted string

See definition from the *Tipper Rx* object. The “Base stations” uuid value should point to itself, while the “Receivers” uuid refers the linked *Tipper Rx* object.

Geoscience INTEGRATOR Objects

List object types specific to INTEGRATOR.

Points

UUID : {6832acf3-78aa-44d3-8506-9574a3510c44}

Not yet geoh5py implemented

To be documented

Microseismic

UUID : {b1388138-5463-11e4-93e8-d3b5f5e17625}

Not yet geoh5py implemented

To be documented

Ground Deformation

UUID : {65a66246-59f7-11e4-aa15-123b93f75cba}

Not yet geoh5py implemented

To be documented

Production Area

UUID : {fc560104-9898-4dbf-9711-07519eb1fc84}

Not yet geoh5py implemented

To be documented

Fixed Plant

UUID : {2dda99b0-9980-4f25-820c-01eb7053b42d}

Not yet geoh5py implemented

To be documented

Blasting

UUID : {20cfa317-e98c-4612-9016-414fb1d9375d}

Not yet geoh5py implemented

To be documented

Mobile Equipment

UUID : {53108442-1664-41ed-99ea-ff4dd273e86c}

Not yet geoh5py implemented

To be documented

Stress

UUID : {60ce697d-59f7-42e0-bb58-88374f1d303a}

Not yet geoh5py implemented

To be documented

Earth Model

UUID : {c4268ef8-6b55-11e4-ab63-ca5fbc5c6e8b}

Not yet geoh5py implemented

To be documented

Mine Model

UUID : {ffd1ae8a-70bc-11e4-bf08-53db6953e95a}

Not yet geoh5py implemented

To be documented

Incidents

UUID : {aca8b138-634e-444c-8698-697b91f4cff9}

Not yet geoh5py implemented

To be documented

Mine infrastructures

UUID : {d15ef4a2-6fc4-40c9-ab3e-11647a81dbe1}

Not yet geoh5py implemented

To be documented

3D Structural Surfaces

UUID : {a69aca26-79d8-4074-bd58-dc2202674071}

Not yet geoh5py implemented

To be documented

3D Domains

UUID : {3ecb7f52-b32c-470d-b2f5-c8b0c2b6dff4}

Not yet geoh5py implemented

To be documented

3D Geological Contact Surfaces

UUID : {46d697f1-50a4-4905-a467-04d5c1e7634c}

Not yet geoh5py implemented

To be documented

Remote Sensing and Air Photos

UUID : {b952c7c5-b636-4f6d-9a59-0cbacd84a332}

Not yet geoh5py implemented

To be documented

Inversions

UUID : {b062ffb4-c57d-49a3-9e96-fa26e7b06e7e}

Not yet geoh5py implemented

To be documented

Topography

UUID : {849635b9-1362-40f1-9edd-f45039ff89ac}

Not yet geoh5py implemented

To be documented

Culture

UUID : {849635b9-1362-40f1-9edd-f45039ff89ac}

Not yet geoh5py implemented

To be documented

Claims, boundaries

UUID : {849635b9-1362-40f1-9edd-f45039ff89ac}

Not yet geoh5py implemented

To be documented

Geophysics

UUID : {80413650-58f0-4c99-94af-48f70affbb65}

Not yet geoh5py implemented

To be documented

Ventilation

UUID : {1cc34f3d-fc50-41d9-8210-d93a73b2c7b4}

Not yet geoh5py implemented

To be documented

Gas Monitoring

UUID : {27e44723-9787-48be-9b0e-67f14d60890b}

Not yet geoh5py implemented

To be documented

Other

UUID : {4ed901bb-0303-43cd-9618-a481f5688844}

Not yet geoh5py implemented

To be documented

Airborne

UUID : {c9f70e63-a30f-428b-bee2-02eed5dde43d}

Not yet geoh5py implemented

To be documented

Ground

UUID : {d9f91038-c7a1-4b72-b3f1-ac7760da16ac}

Not yet geoh5py implemented

To be documented

Borehole

UUID : {0bf977b4-bda8-45d7-9c89-9a41d50849bd}

Not yet geoh5py implemented

To be documented

Neighbourhood Surface

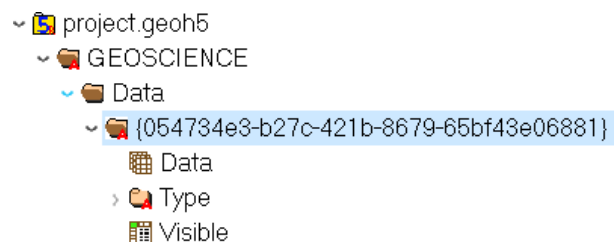
UUID : {88087fb8-76ae-445b-9cdf-68dbce530404}

Not yet geoh5py implemented

To be documented

Data

Containers for data values of various types. Data are currently **always stored as a 1D array**, even in the case of single-value data with the Object association (in which case it is a 1D array of length 1). See the [Data Types](#) section for the list of supported data types.



Attributes

Association

str Describes which part the property is tied to. Must be one of: *Unknown*, *Object*, *Cell*, *Vertex*, *Face* or *Group*

Name

str Name of the data displayed in the project tree.

ID

str Unique identifier (*UUID*) of the group.

Visible

int, 0 or 1 (Optional) Set visible in the 3D camera (checked in the object tree).

Clipping IDs

1D array of UUID (Optional) List of unique identifiers of clipping plane objects.

Allow delete

int, 0 or (default) 1 (Optional) User interface allows deletion.

Allow rename

int, 0 or (default) 1 (Optional) User interface allows renaming.

Public

int, 0 or (default) 1 (Optional) Set accessible in the object tree and other parts of the the user interface.

Types

While they are structured similarly, **each group, object or set of data has a type that defines how its HDF5 datasets should be interpreted.** This type is shared among any number of entities (groups/objects/data sets).

Group Types

Attributes

Name

str Name of the group displayed in the project tree.

ID

str Unique identifier (*UUID*) of the group type.

Description

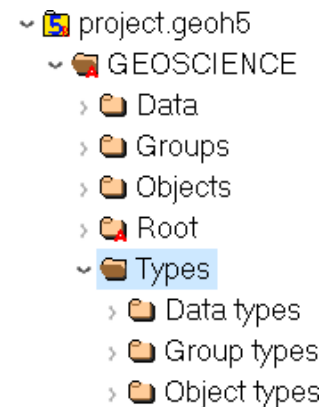
str (Optional) Description of the type.

Allow move contents

int, 0 or (default) 1 (Optional) User interface allows deletion of the content.

Allow delete contents

int, 0 or (default) 1 (Optional) User interface allows deletion of the content.



Object Types

Objects are containers for data values with spatial information.

Attributes

- Name**
str Name of the object displayed in the project tree.
- ID**
str Unique identifier (*UUID*) of the group type.
- Description**
str (Optional) Description of the type.

Data Types

New data types can be created at will by software or users to describe object or group properties. Data of the same type can exist on any number of objects or groups of any type, and each instance can be associated with vertices, cells or the object/group itself. Some data type identifiers can also be reserved as a means of identifying a specific kind of data.

Attributes

- Name**
str Name of the object displayed in the project tree.
- ID**
str Unique identifier (*UUID*) of the data type.

Unlike Groups and Objects, Data entities do not generally have fixed identifier Type. Multiple data entities linked by a type will share common properties (color map, units, etc.). Exceptions to this rule are the fixed:

Geoscience INTEGRATOR Data Set

Microseismic

UUID : {9f9cfec8-3829-11e4-8654-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Ground Deformation

UUID : {c455c010-3829-11e4-9cce-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Production Area

UUID : {a8c95123-349f-4461-b9a4-74f76e659a56}

Not yet geoh5py implemented

To be documented

Blasting

UUID : {f55d8ae4-3829-11e4-a70e-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Stress

UUID : {f8324cdc-3829-11e4-a70e-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Fixed Plant

UUID : {31fc7ef1-3833-11e4-a7fb-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Mobile Equipment

UUID : {75eafc96-3833-11e4-a7fb-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Drillholes & Wells

UUID : {faf65f94-3829-11e4-93fc-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Earth Model Points

UUID : {f38a481f-3833-11e4-a7fb-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Earth Model Regular Grid

UUID : {f1a5e5a6-e651-40dc-b184-7827e792ffb3}

Not yet geoh5py implemented

To be documented

Observation Points

UUID : {d89f963d-16ba-4c6b-87d7-9b95410ab2fb}

Not yet geoh5py implemented

To be documented

Targets

UUID : {528e278e-529c-4d95-b8d1-731cf6ae6b5f}

Not yet geoh5py implemented

To be documented

Anomalies

UUID : {b83ac7a5-5217-4ff1-b0bc-2e8d514655ae}

Not yet geoh5py implemented

To be documented

Fusion Model

UUID : {bc99c8a9-3833-11e4-a7fb-fcddabfddab1}

Not yet geoh5py implemented

To be documented

Samples

UUID : {433ab253-1a73-4414-9159-031cdbf7a9e1}

Not yet geoh5py implemented

To be documented

Geochemistry & Mineralogy

UUID : {72f29283-a4f6-4fc0-a1a8-1417ce5fcbec}

Not yet geoh5py implemented

To be documented

Rock Properties

UUID : {4a067ffb-9d20-46b7-8bd8-a6dde20e8b89}

Not yet geoh5py implemented

To be documented

Incidents

UUID : {016dfd26-7d9b-49a6-97d8-cb31c37e404b}

Not yet geoh5py implemented

To be documented

Mine Infrastructure

UUID : {5e34fb33-86ec-49bb-a3d4-5b21fb158a14}

Not yet geoh5py implemented

To be documented

3D Structural Surfaces

UUID : {0a7fef75-26ba-4e80-9d38-89a76044f908}

Not yet geoh5py implemented

To be documented

3D Domains

UUID : {97909249-8584-40d5-9378-a1fb5b86a3ab}

Not yet geoh5py implemented

To be documented

3D Geological Contact Surfaces

UUID : {c63403e2-b635-4b23-998b-1748fe503f81}

Not yet geoh5py implemented

To be documented

Remote Sensing & Air Photos

UUID : {db53c93a-c57a-4911-92f2-9d0c811268b8}

Not yet geoh5py implemented

To be documented

Inversions

UUID : {57a84d8d-6a33-4dfa-a9f2-66b32e495c7f}

Not yet geoh5py implemented

To be documented

Topography

UUID : {5923bd49-6302-4f8b-963a-cba57ac757ae}

Not yet geoh5py implemented

To be documented

Culture

UUID : {bbccf928-d410-4d59-b737-4b4c1f8c84ca}

Not yet geoh5py implemented

To be documented

Claims, boundaries

UUID : {1bcb5c4-c33a-4682-ac47-88694ca67905}

Not yet geoh5py implemented

To be documented

Geophysics

UUID : {9b097cc1-66cb-4088-83dd-c447cba542df}

Not yet geoh5py implemented

To be documented

Ventilation

UUID : {b716d06a-8104-4086-a029-b10d1a545b49}

Not yet geoh5py implemented

To be documented

Gas Monitoring

UUID : {844354fa-41ae-416c-b33f-bf5bfbedc8f5}

Not yet geoh5py implemented

To be documented

Other

UUID : {7bebe936-2e04-4bd6-b050-b128ec5c078d}

Not yet geoh5py implemented

To be documented

Primitive type

str

Specifies the kind of data values stored as HDF5 dataset. Must be one of:

Primitive Types

To be further documented

Float

- Stored as a 1D array of 32-bit float type
- No data value: 1.175494351e-38 (FLT_MIN, considering use of NaN)

Integer

- Stored as a 1D array of 32-bit integer type
- No data value: -2147483648 (INT_MIN, considering use of NaN)

Text

- Stored as a 1D array of UTF-8 encoded, variable-length string type
- No data value : empty string

Referenced

- Stored as a 1D array of 32-bit unsigned integer type (native)
- Value map : (1D composite type array dataset - Key (unsigned int), Value (variable-length utf8 string)) must exist under type
- No data value : 0 (key is tied to value “Unknown”)

DateTime

- Stored as a 1D array of variable-length strings formatted according to the [ISO 8601](#) extended specification for representations of UTC dates and times (Qt implementation), taking the form YYYY-MM-DDTHH:mm:ss[Z][+/-]HH:mm]
- No data value : empty string

Filename

- Stored as a 1D array of UTF-8 encoded, variable-length string type designating a file name
- For each file name within “Data”, an opaque dataset named after the filename must be added under the Data instance, containing a complete binary dump of the file
- Different files (under the same object/group) must be saved under different names
- No data value : empty string

Blob

- Stored as a 1D array of 8-bit char type (native) (value '0' or '1')
- For each index set to 1, an opaque dataset named after the index (e.g. "1", "2", etc) must be added under the Data instance, containing the binary data tied to that index
- No data value : 0

Description

str (Optional) Description of the type.

Units

str (Optional) Data units

Color map

1D compound array

[*Value* double, *Red* uint, *Green* uint, *Blue* uint, *Alpha* uint]

(Optional) Records colors assigned to value ranges. The *Value* mark the start of the range)

Value map

(1D compound array dataset)

[*Key* uint, *Value* str]

Required only for reference data types (classifications)

Transparent no data

int, 0 or (default) 1 (Optional) Whether or not absence of data/filtered data should be hidden in the viewport.

Hidden

int, 0 or (default) 1 (Optional) Whether or not the data type should appear in the data type list.

Scientific notation

int, 0 or (default) 1 (Optional) Whether or not the data values of this type should be displayed in scientific notation.

Precision

int (Optional) The number of decimals (or significant digits in case of scientific notation) used when displayed data values of this type.

Number of bins

int, default=50 (Optional) Number of bins used when displaying histogram

Duplicate type on copy

int, 0 or (default) 1 (Optional) When enabled, a separate copy of this data type will be created and used when data of this type is copied.

2.4.3 Standards

General notes on formatting.

- All text data and attributes are variable-length and use UTF-8 encoding
- All numeric data uses INTEL PC native types
- Boolean values are stored using char (0:false, 1:true)
- Anything found in a geoh5 v1.0 file which is not mentioned in this document is optional information

2.4.4 External Links

- [HDFView](#).
- [Precompiled binaries for multiple platforms](#)
- **Libraries for accessing HDF5 data**
 - C, C, .NET
 - Python
 - Matlab

2.5 UI.JSON Format

2.5.1 About

The **ui.json** format provides a schema to create a simple User Interface (UI) between geoh5py and [Geoscience ANALYST Pro](#). The format uses [JSON objects](#) to represent *script parameters* used in the UI, and pass those parameters to an accompanying python script.

Each ui.json object requires at least a **label** and **value** member, however additional members can be used to define different types of input and additional dependencies between parameters.

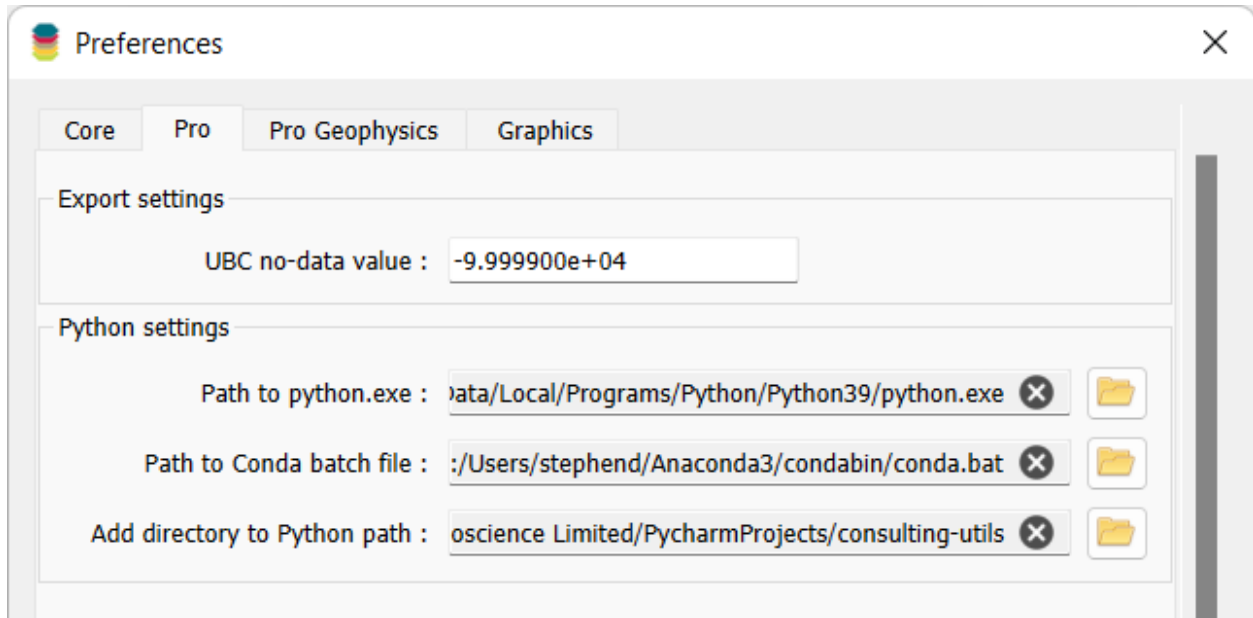
For example, a simple ui.json below describes a single parameter called 'grid_object', which is used to select a block model within a geoh5 file.

```
{
  "grid_object": {
    "meshType": [{"B020A277-90E2-4CD7-84D6-612EE3F25051"}],
    "main": true,
    "label": "Select Block Model",
    "value": ""
  }
}
```

Note: The **meshType** used to select the grid object is defined by a list of UUID. A complete list of UUID's for geoh5 object types are available in the [geoh5 objects documentation](#).

2.5.2 Usage with Geoscience ANALYST Pro

A `ui.json` file contains the parameters that reference a Python script to run. Geoscience ANALYST must be configured prior to running the application by setting the path to Python scripts or to a Conda environment as defined by the `conda_environment` parameter.



`run_command str`

Name of python script excluding the `.py` extension (i.e., “run_me” for `run_me.py`) required for Geoscience ANALYST Pro to run on save or auto-load.

`conda_environment str`

Optional name of conda environment to activate when running the python script in `run_command`

`title str`

Title of user interface window

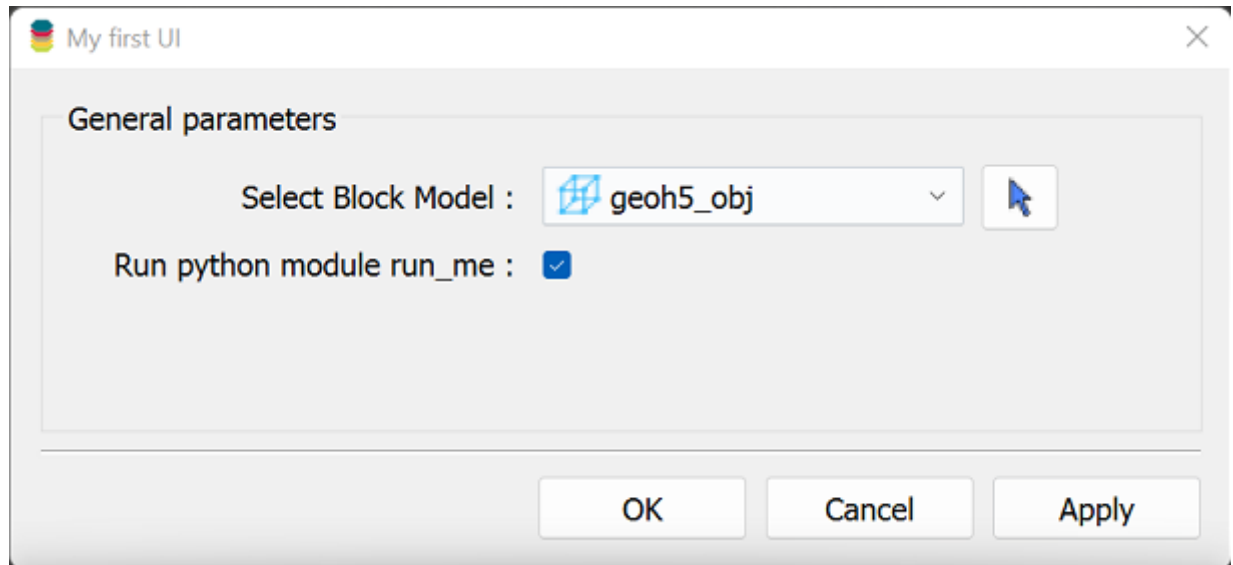
To complete the example above, add the `run_command`, and `title` parameter to the `ui.json` file.

```
{
  "grid_object": {
    "meshType": [{"B020A277-90E2-4CD7-84D6-612EE3F25051"}],
    "main": true,
    "label": "Select Block Model",
    "value": ""
  },
  "title": "My first UI",
  "run_command": "run_me"
}
```

Within the accompanying python script, the parameters from the `ui.json` is passed to the python script as a dictionary of arguments, and can be accessed using the `InputFile` module of `geoh5py` as shown below:

```
import sys
from geoh5py.ui_json import InputFile
```

(continues on next page)



(continued from previous page)

```

ui_json = sys.argv[1]
ifile = InputFile.read_ui_json(ui_json)
params = ifile.data

# Get the block model grid object
bm = params["grid_object"]
print(f"The selected object name is {bm.name}")

```



When a **ui.json** is run within Geoscience ANALYST Pro, the following parameters are updated or added:

- The **value** member is updated with the UUID of the object selected in the UI
- The **enabled** member `bool` is set for whether the parameter is enabled
- The *Data parameter* will also have updated **isValue** and **property** members. The **isValue** `bool` member is *true* if the **value** member was selected and *false* if the **property** member was selected.

The following JSON objects will be written (and overwritten if given) upon running a `ui.json` from Geoscience ANALYST Pro:

- **monitoring_directory** `str` the absolute path of a monitoring directory. Workspace files written to this folder will be automatically processed by Geoscience ANALYST.
- **workspace_geoh5** `str` (Optional) Path to the source geoh5 file (for reference only)
- **geoh5** `str` the absolute path to the geoh5 written containing all the objects of the workspace within the parameters of the **ui.json**. One only needs to use this workspace along with the JSON file to access the objects with `geoh5py`.

2.5.3 Parameters available for all ui.json objects

The following members are available to all input parameters in the ui.json schema.

label str

Required string describing parameter. A colon will automatically be added within Geoscience ANALYST, so this should be omitted.

value str, int, bool , or float

This required member takes a different form, including the empty string "", depending on the *parameter type*. The value is updated when written from Geoscience ANALYST.

main bool

If set to true, the parameter is shown in the first tab of the UI and will throw an error if not present (and not optional). Optional parameters may be set to main. When main is not given or is false, the parameter will be under the *Optional Parameters* tab.

tooltip str

String describing the parameter in detail that appears when the mouse hovers over it.

optional bool

true or *false* on whether the parameter is optional. On output, check if *enabled* is set to true.

enabled bool

true or *false* if the parameter is enabled. The default is true. If a parameter is optional and not enabled, it will start as disabled (grey and inactive in the UI).

group str

Name of the group to which the parameter belongs. Adds a box and name around the parameters with the same case-sensitive group name.

groupOptional bool

If true, adds a checkbox in the top of the group box next to the name. The group parameters will be disabled if not checked. The initial state depends on the **groupDependency** and **groupDependencyType** members and the **enabled** member of the group's parameters.

dependency str

The name of the parameter which this parameter is dependent upon. The dependency parameter should be optional or boolean parameter (i.e., has a checkbox).

dependencyType str

What happens when the dependency member is checked. Options are *enabled* or *disabled*

groupDependency str

The name of the object of which the group of the parameter is dependent upon. This member will also require the **groupOptional** member to be present and set to *true*. Be sure that the object is not within the group.

groupDependencyType str

What happens when the group's dependency parameter is checked. Options are *enabled* or *disabled*.

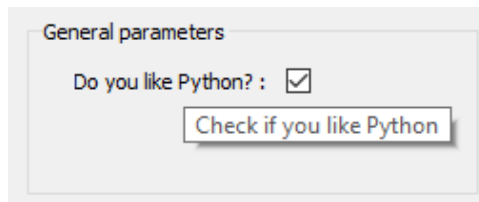
2.5.4 Additional Parameters

The following sections define different object specific parameters that can be used in the **ui.json** schema.

Boolean Parameter

A parameter named “input” that has a bool value.

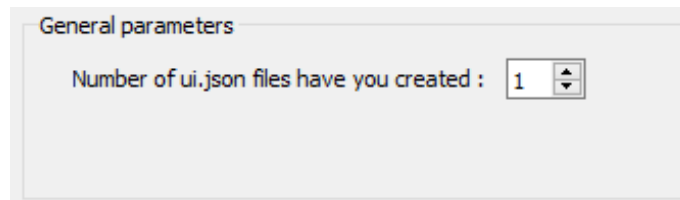
```
{
  "input":{
    "main": true,
    "label": "Do you like Python?",
    "value": true,
    "tooltip": "Check if you like Python"
  }
}
```

A screenshot of a UI widget titled "General parameters". It contains a label "Do you like Python? :" followed by a checked checkbox. Below the checkbox is a button with the text "Check if you like Python".

Integer Parameter

A parameter that has an int value. The optional parameters min and max invoke a validator to insure the bound(s) are enforced.

```
{
  "file_xp":{
    "main": true,
    "label": "Number of ui.json files have you created",
    "value": 1,
    "min": 0,
    "max": 100
  }
}
```

A screenshot of a UI widget titled "General parameters". It contains a label "Number of ui.json files have you created : " followed by a text input field containing the number "1" and a vertical spinner control.

Float Parameter

A parameter that has a `float` value. The optional parameters are:

min float

Minimum value allowed for validator of the **value** member. The default is the minimum numeric limits of float.

max float

Maximum value allowed for validator of the **value** member. The default is the maximum numeric limits of float.

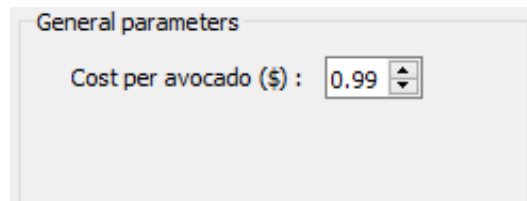
lineEdit bool

Boolean whether to use a line edit (**true**) or a spin box (**false**). The default is true.

precision int

Number of decimal places in the line edit or spin box

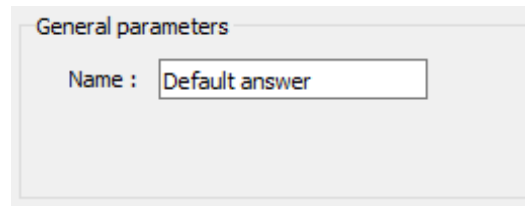
```
{
  "avacado": {
    "main": true,
    "label": "Cost per avocado ($)",
    "value": 0.99,
    "min": 0.29,
    "precision": 2,
    "lineEdit": false,
    "max": 2.79
  }
}
```



String Parameter

For a simple string parameter, use an empty `str` value to have an empty string. Only a `label` and `value` is required.

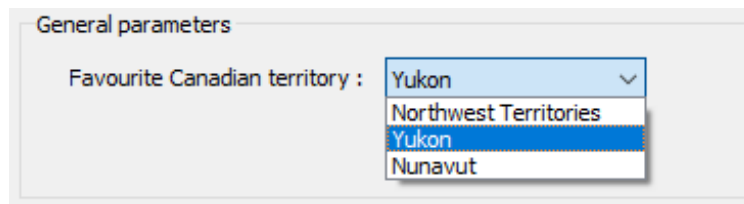
```
{
  "my_string": {
    "main": true,
    "label": "Name",
    "value": "Default answer"
  }
}
```



Multi-choice string Parameter

For a drop-down selection, add a `choiceList` member with an array of strings (`str`)

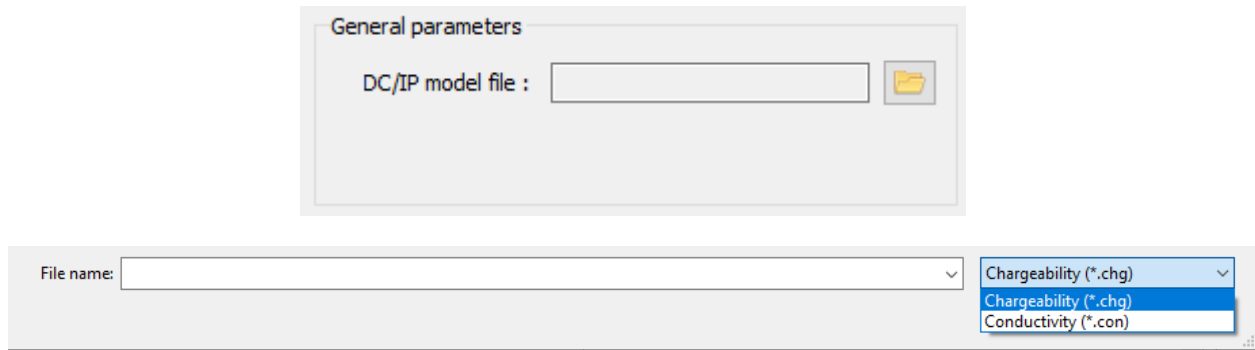
```
{
  "favourites": {
    "choiceList": ["Northwest Territories",
                  "Yukon",
                  "Nunavut"],
    "main": true,
    "label": "Favourite Canadian territory",
    "value": "Yukon"
  }
}
```



File Parameter

A file parameter comes with an icon to choose the file, with a `str` value. Extra members of the file object parameter are **fileDescription** and **fileType**. Both of these are `str` types and can be arrays, but must be of the same length

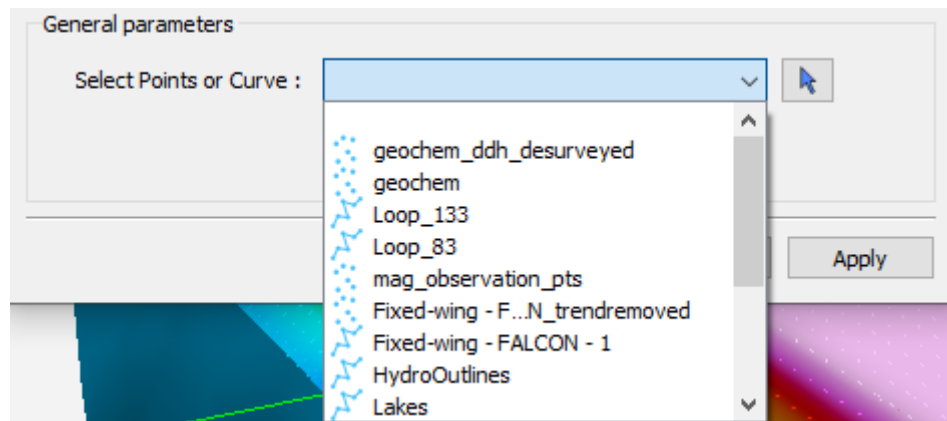
```
{
  "model_file": {
    "fileDescription": ["Chargeability", "Conductivity"],
    "fileType": ["chg", "con"],
    "main": true,
    "label": "DC/IP model file",
    "value": ""
  }
}
```



Geoscience ANALYST Object Parameter

To choose an object from a dropdown menu, the [universally unique identifier \(UUID\)](#) of the *Object Type*: is required for the filtering of objects. This is given as a single or array of `str` in the member **meshType**. The icon to pick the object comes with this parameter. The value returned is the *UUID* of the Geoscience ANALYST object selected.

```
{
  "interesting_object": {
    "meshType": ["{202C5DB1-A56D-4004-9CAD-BAAFD8899406}" ,
      "{6A057FDC-B355-11E3-95BE-FD84A7FFCB88}"],
    "main": true,
    "label": "Select Points or Curve",
    "value": ""
  }
}
```



Geoscience ANALYST Data parameter

Creating a parameter to choose a Geoscience ANALYST object's data requires extra members:

dataType str

Describes the type of data to filter. One or more (as an array) of these key words: Integer, Float, Text, Referenced, Vector, DateTime, Geometric, Boolean, or Text.

association str

Describes the geometry of the data. One or more of these key words: Vertex, Cell, or Face.

parent str

Either a *UUID* of the parent or the name of the *Object parameter* JSON object to allow the user to choose the mesh.

isValue bool

Describes whether to read the **value** (float) or **property** (str) member. If not given, the value member is an *UUID* and is considered a *drop-down data parameter*. If this member is given along with **property**, then an icon is added to the UI element, which switches between the **value** (line edit) and **property** (drop-down) choices. This value is updated on export depending on the style choice (float or str)

property str.

Data *UUID* that is selected when **isValue** is present. Geoscience ANALYST Pro will update this value on export.

min float

Optional minimum value allowed for validator of the **value** member. The default is the minimum numeric limits of float.

max float

Optional maximum value allowed for validator of the **value** member. The default is the maximum numeric limits of float.

precision int

Optional number of decimal places for the value.

Drop-down Parameter

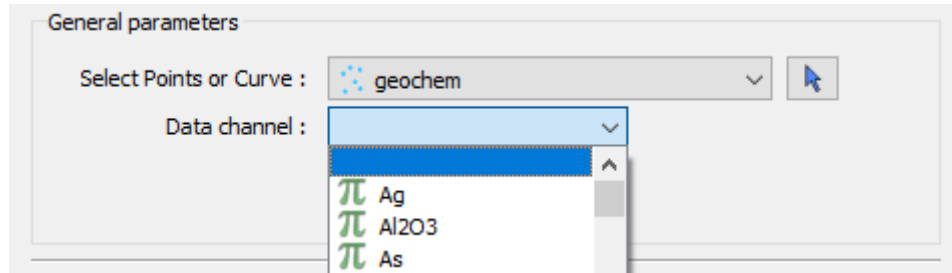
In this example, the object parameter *data_mesh* is also given for reference.

```
{
  "data_channel": {
    "main": true,
    "association": "Vertex",
    "dataType": "Float",
    "label": "Data channel",
    "parent": "data_mesh",
    "value": ""
  },
  "data_mesh": {
    "main": true,
    "meshType": ["{202C5DB1-A56D-4004-9CAD-BAAFD8899406}" ,
      "{6A057FDC-B355-11E3-95BE-FD84A7FFCB88}"],
    "main": true,
    "label": "Select Points or Curve",
```

(continues on next page)

(continued from previous page)

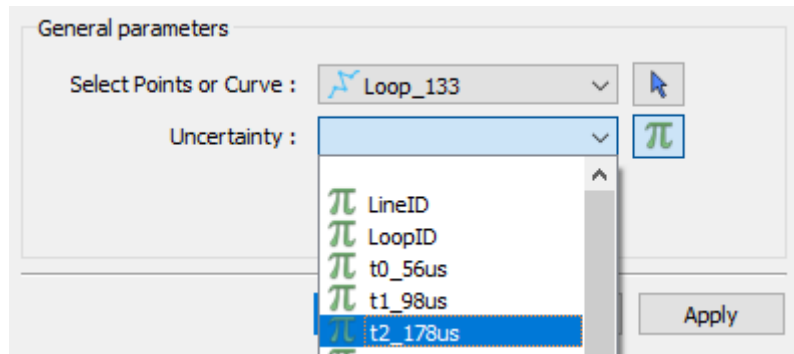
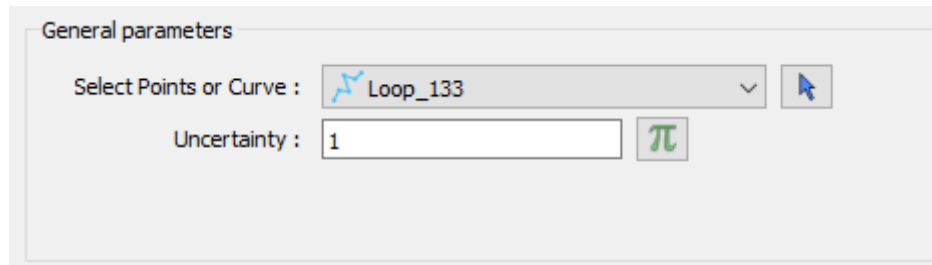
```
"value": ""
}
}
```



Data or value Parameter

In some cases, a parameter may take its data from a Geoscience ANALYST object or simply a float value. The use of the member **isValue** and **property** together allows for the UI to switch between these two cases. In the top image, the **isValue** is true, so the **value** member of 1.0 will initially be active. When the icon is clicked, the type of input is switched to the **property** member (bottom image). The **uncertainty channel** object also depends on the **data_mesh** object. The drop-down selection will filter data from the chosen object that is located on the vertices and is float. The **isValue** is set to false upon export in this case.

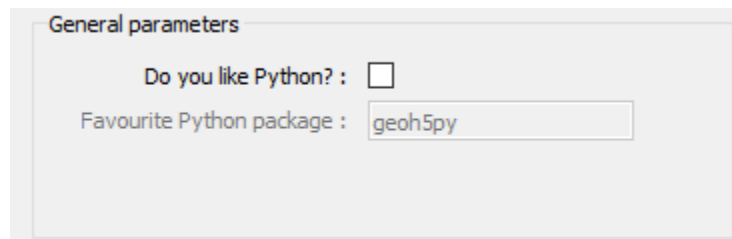
```
{
  "uncertainty_channel": {
    "main": true,
    "association": "Vertex",
    "dataType": "Float",
    "isValue": true,
    "property": "",
    "min": 0.001,
    "label": "Uncertainty",
    "parent": "data_mesh",
    "value": 1.0
  },
  "data_mesh": {
    "main": true,
    "meshType": ["{202C5DB1-A56D-4004-9CAD-BAAFD8899406}" ,
      "{6A057FDC-B355-11E3-95BE-FD84A7FFCB88}"],
    "main": true,
    "label": "Select Points or Curve",
    "value": ""
  }
}
```



Dependencies on other parameters

Use the **dependency** and **dependencyType** members to create dependencies. The parameter driving the dependency should set **optional** to true or be a *Boolean parameter*. Below are a couple of examples. The first initializes the *favourite_package* parameter as disabled until the *python_interest* parameter is checked. The second shows the opposite when the **enabled** member is set to true.

```
{
  "python_interest": {
    "main": true,
    "label": "Do you like Python?",
    "value": false,
    "tooltip": "Check if you like Python"
  },
  "favourite_package": {
    "main": true,
    "label": "Favourite Python package",
    "value": "geoh5py",
    "dependency": "python_interest",
    "dependencyType": "enabled"
  }
}
```



The next example has a dependency on an optional parameter. The **enabled** member is set to false so

that it is not automatically checked. The *city* and *territory* parameters will be enabled when the *territory* checkbox is checked.

```
{
  "territory": {
    "choiceList": ["Northwest Territories",
                  "Yukon",
                  "Nunavut"],
    "main": true,
    "label": "Favourite Canadian territory",
    "value": "Yukon",
    "optional": true,
    "enabled": false
  },
  "city": {
    "main": true,
    "choiceList": ["Yellowknife",
                  "Whitehorse",
                  "Iqaluit"],
    "label": "Favourite capital",
    "value": "",
    "dependency": "territory",
    "dependencyType": "enabled"
  }
}
```

General parameters

☐ Favourite Canadian territory : Yukon ▼

Favourite capital : Yellowknife ▼

General parameters

☒ Favourite Canadian territory : Yukon ▼

Favourite capital : Iqaluit ▼

2.5.5 Tips on creating UIs

- Keep labels concise
- Write detailed tooltips
- Group related attributes
- Don't include the **main** member with every parameter. “Non-main” members are designated to a second page under *Optional parameters*
- Utilize **optional** object members and dependencies.

2.5.6 External Links

- [JSON Schema](#)
- [Universally Unique Identifier \(UUID\)](#)

2.6 Release Notes

2.6.1 Release 0.6.0 - 2023/01/26

- GEOPY-700, 701, 721, 726: Add functionality to convert between Grid2D and GeoImages.
- GEOPY-843: Update drillhole group compatibility with ANALYST v4.2
- GEOPY-746: Implement ground TEM (large-loop) survey type.

2.6.2 Release 0.5.0 - 2022/10/26

- GEOPY-624: Add functionality to remove vertices and cells
- GEOPY-644: Functionality to copy object within box extent. Only implemented for vertex-based object.
- **Bug fixes:**
 - GEOPY-650: Deal with INTEGRATOR text data in byte format.
 - GEOPY-615: Fix de-survey method for older geoh5 v1 format.

2.6.3 Release 0.4.0 - 2022/08/26

Major release adding new classes and fixing issues with the DrillholeGroup class.

- **Fixes for concatenated DrillHoleGroup**
 - GEOPY-598: Implement IntegratorDrillholeGroup class
 - GEOPY-583: Better handling of adding and removing concatenated drillholes and data intervals.
- GEOPY-584: Preserve integer values on IntegerData class.
- GEOPY-548: Allow TextData values on vertices and cells.
- GEOPY-329: API implementation of DrapeModel object class.
- GEOPY-462: Documentation fixes

2.6.4 Release 0.3.1 - 2022/08/26

This release addresses issues encountered after the 0.3.0 release.

- GEOPY-608: Check for 'allow_delete' status before removing.
- GEOPY-600: Fix crash on missing 'Group types' group from project written by ANALYST.
- GEOPY-587: Increase PEP8 compliance after pylint update.
- GEOPY-575: Improve ui.json documentation.

2.6.5 Release 0.3.0 - 2022/06/30

This release addresses changes introduced by the geoh5 v2.0 standard.

- Drillhole objects and associated data are stored as Concatenated entities under the DrillholeGroup.
- Use of context manager for the Workspace with options for read/write mode specifications added.
- Implementation of a SimPEGGroup entity.

2.6.6 Release 0.2.0 - 2022/04/18

- Add MT, tipper and airborne time-domain survey objects.
- Add ui.json read/write with validations
- Bug fixes and documentation.

2.6.7 Release 0.1.6 - 2021/12/09

- Fix StatsCache on value changes.
- Fix crash if data values are None.
- Clean up for linters

2.6.8 Release 0.1.5 - 2021/11/05

- Fix for copying of direct-current survey.
- Fix documentation.

2.6.9 Release 0.1.4 - 2021/08/31

- Add direct_current survey type and related documentation.
- Fix for drillholes with single survey location anywhere along the borehole.
- Fix for entity.parent setter. Changes are applied directly to the target workspace.
- Improve Typing.

2.7 Feedback

Have comments or suggestions? Submit feedback. All the content can be found on our [github](#) repository.

2.7.1 Contributors

- [Mira Geoscience](#)

PYTHON MODULE INDEX

g

geoh5py, 111

geoh5py.data, 52

geoh5py.data.blob_data, 46

geoh5py.data.color_map, 46

geoh5py.data.data, 46

geoh5py.data.data_association_enum, 47

geoh5py.data.data_type, 47

geoh5py.data.data_unit, 48

geoh5py.data.datetime_data, 49

geoh5py.data.filename_data, 49

geoh5py.data.float_data, 49

geoh5py.data.geometric_data_constants, 49

geoh5py.data.integer_data, 50

geoh5py.data.numeric_data, 50

geoh5py.data.primitive_type_enum, 50

geoh5py.data.reference_value_map, 51

geoh5py.data.referenced_data, 51

geoh5py.data.text_data, 51

geoh5py.data.unknown_data, 52

geoh5py.groups, 57

geoh5py.groups.container_group, 52

geoh5py.groups.custom_group, 52

geoh5py.groups.drillhole_group, 52

geoh5py.groups.gifttools_group, 53

geoh5py.groups.group, 53

geoh5py.groups.group_type, 54

geoh5py.groups.integrator_group, 54

geoh5py.groups.maps_group, 56

geoh5py.groups.notype_group, 56

geoh5py.groups.property_group, 56

geoh5py.groups.root_group, 57

geoh5py.groups.simpeg_group, 57

geoh5py.groups.survey_group, 57

geoh5py.io, 64

geoh5py.io.h5_reader, 57

geoh5py.io.h5_writer, 60

geoh5py.objects, 84

geoh5py.objects.block_model, 72

geoh5py.objects.curve, 73

geoh5py.objects.drape_model, 73

geoh5py.objects.drillhole, 74

geoh5py.objects.geo_image, 76

geoh5py.objects.grid2d, 77

geoh5py.objects.integrator, 79

geoh5py.objects.label, 79

geoh5py.objects.notype_object, 79

geoh5py.objects.object_base, 79

geoh5py.objects.object_type, 82

geoh5py.objects.octree, 82

geoh5py.objects.points, 83

geoh5py.objects.surface, 84

geoh5py.objects.surveys, 72

geoh5py.objects.surveys.direct_current, 70

geoh5py.objects.surveys.electromagnetics, 70

geoh5py.objects.surveys.electromagnetics.airborne_tem,
64

geoh5py.objects.surveys.electromagnetics.base,
66

geoh5py.objects.surveys.electromagnetics.magnetotellurics,
68

geoh5py.objects.surveys.electromagnetics.tipper,
69

geoh5py.objects.surveys.magnetics, 71

geoh5py.shared, 97

geoh5py.shared.concatenation, 84

geoh5py.shared.entity, 87

geoh5py.shared.entity_type, 90

geoh5py.shared.exceptions, 90

geoh5py.shared.utils, 92

geoh5py.shared.validators, 94

geoh5py.shared.weakref_utils, 96

geoh5py.ui_json, 105

geoh5py.ui_json.constants, 97

geoh5py.ui_json.input_file, 97

geoh5py.ui_json.templates, 99

geoh5py.ui_json.utils, 103

geoh5py.ui_json.validation, 105

geoh5py.workspace, 111

geoh5py.workspace.workspace, 106

INDEX

A

- `ab_cell_id` (`geoh5py.objects.surveys.direct_current.PotentialElectrode` property), 71
- `ab_map` (`geoh5py.objects.surveys.direct_current.PotentialElectrode` property), 71
- `activate()` (`geoh5py.workspace.workspace.Workspace` method), 106
- `active()` (`geoh5py.workspace.workspace.Workspace` static method), 106
- `active_workspace()` (in module `geoh5py.workspace.workspace`), 111
- `add_children()` (`geoh5py.shared.entity.Entity` method), 87
- `add_comment()` (`geoh5py.groups.group.Group` method), 53
- `add_comment()` (`geoh5py.objects.object_base.ObjectBase` method), 79
- `add_components_data()` (`geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey` method), 66
- `add_data()` (`geoh5py.objects.drillhole.Drillhole` method), 74
- `add_data()` (`geoh5py.objects.object_base.ObjectBase` method), 79
- `add_data_to_group()` (`geoh5py.objects.object_base.ObjectBase` method), 80
- `add_default_ab_cell_id()` (`geoh5py.objects.surveys.direct_current.CurrentElectrode` method), 70
- `add_file()` (`geoh5py.data.data.Data` method), 46
- `add_file()` (`geoh5py.shared.entity.Entity` method), 87
- `add_save_concatenated()` (`geoh5py.shared.concatenation.Concatenator` method), 86
- `add_validate_component_data()` (`geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey` method), 67
- `add_vertices()` (`geoh5py.objects.drillhole.Drillhole` method), 74
- `AirborneGeophysics` (class in `geoh5py.groups.survey_group`), 57
- `AirborneMagnetics` (class in `geoh5py.objects.surveys.magnetics`), 71
- `AirborneTEMReceivers` (class in `geoh5py.objects.surveys.electromagnetics.airborne_tem`), 64
- `AirborneTEMTransmitters` (class in `geoh5py.objects.surveys.electromagnetics.airborne_tem`), 64
- `AirborneTheme` (class in `geoh5py.groups.integrator_group`), 54
- `allow_delete` (`geoh5py.shared.entity.Entity` property), 87
- `allow_delete_content` (`geoh5py.groups.group_type.GroupType` property), 54
- `allow_move` (`geoh5py.shared.entity.Entity` property), 88
- `allow_move_content` (`geoh5py.groups.group_type.GroupType` property), 54
- `allow_rename` (`geoh5py.shared.entity.Entity` property), 88
- `as_str_if_utf8_bytes()` (in module `geoh5py.shared.utils`), 92
- `as_str_if_uuid()` (in module `geoh5py.shared.utils`), 92
- `association` (`geoh5py.data.data.Data` property), 46
- `association` (`geoh5py.groups.property_group.PropertyGroup` property), 56
- `association_validator` (`geoh5py.ui_json.input_file.InputFile` attribute), 97
- `AssociationValidationError`, 90
- `AssociationValidator` (class in `geoh5py.shared.validators`), 94
- `AtLeastOneValidationError`, 90
- `AtLeastOneValidator` (class in `geoh5py.shared.validators`), 94
- `attribute_map` (`geoh5py.groups.property_group.PropertyGroup` property), 56
- `attribute_map` (`geoh5py.shared.entity.Entity` property), 88
- `attribute_map` (`geoh5py.shared.entity_type.EntityType` property), 90
- `attribute_map` (`geoh5py.workspace.workspace.Workspace`

property), 106

attributes_keys (geoh5py.shared.concatenation.Concatenator
property), 86

B

base_refine() (geoh5py.objects.octree.Octree
method), 82

base_stations (geoh5py.objects.surveys.electromagnetics.tipper.BaseTipper
property), 69

BaseAirborneTEM (class in
geoh5py.objects.surveys.electromagnetics.airborne_tem),
65

BaseEMSurvey (class in
geoh5py.objects.surveys.electromagnetics.base),
66

BaseTEMSurvey (class in
geoh5py.objects.surveys.electromagnetics.base),
68

BaseTipper (class in geoh5py.objects.surveys.electromagnetics.tipper),
69

BaseValidationError, 91

BaseValidator (class in geoh5py.shared.validators), 94

BLOB (geoh5py.data.primitive_type_enum.PrimitiveTypeEnum
attribute), 50

BlobData (class in geoh5py.data.blob_data), 46

BlockModel (class in geoh5py.objects.block_model), 72

bool_parameter() (in module
geoh5py.ui_json.templates), 99

bool_value() (in module geoh5py.shared.utils), 92

C

CELL (geoh5py.data.data_association_enum.DataAssociationEnum
attribute), 47

cell_center_u (geoh5py.objects.grid2d.Grid2D prop-
erty), 77

cell_center_v (geoh5py.objects.grid2d.Grid2D prop-
erty), 77

cell_delimiters (geoh5py.objects.block_model.BlockModel
property), 72

cells (geoh5py.objects.curve.Curve property), 73

cells (geoh5py.objects.drillhole.Drillhole property), 74

cells (geoh5py.objects.geo_image.GeoImage property),
76

cells (geoh5py.objects.object_base.ObjectBase prop-
erty), 80

cells (geoh5py.objects.surface.Surface property), 84

centroids (geoh5py.objects.block_model.BlockModel
property), 72

centroids (geoh5py.objects.drape_model.DrapeModel
property), 73

centroids (geoh5py.objects.grid2d.Grid2D property),
78

centroids (geoh5py.objects.octree.Octree property), 82

channels (geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey
property), 67

check_vector_length()
(geoh5py.data.numeric_data.NumericData
method), 50

children (geoh5py.shared.entity.Entity property), 88

choice_string_parameter() (in module
geoh5py.ui_json.templates), 99

clear_stats_cache() (geoh5py.io.h5_writer.H5Writer
class method), 60

clip_by_extent() (geoh5py.objects.drape_model.DrapeModel
method), 73

clip_by_extent() (geoh5py.objects.object_base.ObjectBase
method), 80

clip_by_extent() (geoh5py.objects.points.Points
method), 83

clipping_ids (geoh5py.shared.entity.Entity property),
88

close() (geoh5py.workspace.workspace.Workspace
method), 106

collar (geoh5py.objects.drillhole.Drillhole property),
74

collect() (in module geoh5py.ui_json.utils), 103

color_map (geoh5py.data.data_type.DataType prop-
erty), 47

ColorMap (class in geoh5py.data.color_map), 46

comments (geoh5py.groups.group.Group property), 53

comments (geoh5py.objects.object_base.ObjectBase
property), 80

CommentsData (class in geoh5py.data.text_data), 51

compare_entities() (in module geoh5py.shared.utils),
92

components (geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey
property), 67

compute_deviation() (in module
geoh5py.objects.drillhole), 75

concat_attr_str (geoh5py.shared.concatenation.Concatenated
property), 84

concat_attr_str (geoh5py.shared.concatenation.Concatenator
property), 86

Concatenated (class in geoh5py.shared.concatenation),
84

concatenated_attributes
(geoh5py.shared.concatenation.Concatenator
property), 86

concatenated_object_ids
(geoh5py.shared.concatenation.Concatenator
property), 86

ConcatenatedData (class in
geoh5py.shared.concatenation), 84

ConcatenatedDrillhole (class in
geoh5py.shared.concatenation), 84

ConcatenatedObject (class in
geoh5py.shared.concatenation), 85

ConcatenatedPropertyGroup (class in method), 107
geoh5py.shared.concatenation), 85
Concatenator (class in geoh5py.shared.concatenation), 85
concatenator (geoh5py.shared.concatenation.Concatenated property), 84
container_group2name() (in module geoh5py.ui_json.utils), 103
ContainerGroup (class in geoh5py.groups.container_group), 52
contributors (geoh5py.workspace.workspace.Workspace property), 106
converter (geoh5py.objects.object_base.ObjectBase property), 80
copy() (geoh5py.objects.surveys.direct_current.CurrentElectrode method), 70
copy() (geoh5py.objects.surveys.direct_current.PotentialElectrode method), 71
copy() (geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey method), 67
copy() (geoh5py.shared.concatenation.Concatenator method), 86
copy() (geoh5py.shared.entity.Entity method), 88
copy_from_extent() (geoh5py.groups.group.Group method), 53
copy_from_extent() (geoh5py.objects.object_base.ObjectBase method), 80
copy_property_groups() (geoh5py.workspace.workspace.Workspace class method), 106
copy_to_parent() (geoh5py.workspace.workspace.Workspace method), 106
cost (geoh5py.objects.drillhole.Drillhole property), 74
create() (geoh5py.data.data_type.DataType class method), 47
create() (geoh5py.shared.entity.Entity class method), 88
create_custom() (geoh5py.groups.group_type.GroupType static method), 54
create_custom() (geoh5py.objects.object_type.ObjectType static method), 82
create_data() (geoh5py.workspace.workspace.Workspace method), 106
create_dataset() (geoh5py.io.h5_writer.H5Writer class method), 61
create_entity() (geoh5py.workspace.workspace.Workspace method), 107
create_from_concatenation() (geoh5py.workspace.workspace.Workspace method), 107
create_geoh5() (geoh5py.io.h5_writer.H5Writer class method), 61
create_object_or_group() (geoh5py.workspace.workspace.Workspace
crossline_offset (geoh5py.objects.surveys.electromagnetics.airborne_tem.BaseAirborneTEM property), 65
current_electrodes (geoh5py.objects.surveys.direct_current.CurrentElectrode property), 70
current_electrodes (geoh5py.objects.surveys.direct_current.PotentialElectrode property), 71
current_line_id (geoh5py.objects.curve.Curve property), 73
CurrentElectrode (class in geoh5py.objects.surveys.direct_current), 70
Curve (class in geoh5py.objects.curve), 73
CustomGroup (class in geoh5py.groups.custom_group), 52
D
Data (class in geoh5py.data.data), 46
Data (geoh5py.shared.concatenation.Concatenator property), 86
data (geoh5py.ui_json.input_file.InputFile property), 97
data (geoh5py.workspace.workspace.Workspace property), 107
data_parameter() (in module geoh5py.ui_json.templates), 99
data_value_parameter() (in module geoh5py.ui_json.templates), 100
DataAssociationEnum (class in geoh5py.data.data_association_enum), 47
DataType (class in geoh5py.data.data_type), 47
DataUnit (class in geoh5py.data.data_unit), 48
DATETIME (geoh5py.data.primitive_type_enum.PrimitiveTypeEnum attribute), 50
DatetimeData (class in geoh5py.data.datetime_data), 49
deactivate() (geoh5py.workspace.workspace.Workspace method), 107
default_collocation_distance (geoh5py.objects.drillhole.Drillhole property), 75
default_input_types (geoh5py.objects.surveys.electromagnetics.airborne_tem.BaseAirborneTEM property), 65
default_input_types (geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey property), 67
default_input_types (geoh5py.objects.surveys.electromagnetics.base.BaseTEMSurvey property), 68
default_input_types (geoh5py.objects.surveys.electromagnetics.magnetotellurics.MTResistivity property), 68
default_input_types (geoh5py.objects.surveys.electromagnetics.tipper.BaseTipper property), 69

`default_type_uid()` (`geoh5py.objects.surveys.electromagnetics.airborneTEMReceivers` class method), 64
`default_type_uid()` (`geoh5py.objects.surveys.electromagnetics.airborneTEMTransmitters` class method), 65
`default_type_uid()` (`geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey` class method), 67
`default_type_uid()` (`geoh5py.objects.surveys.electromagnetics.grid2d.Grid2D` class method), 69
`default_type_uid()` (`geoh5py.objects.surveys.electromagnetics.magnetotellurics.MTReceivers` class method), 70
`default_type_uid()` (`geoh5py.objects.surveys.electromagnetics.magnetotellurics.MTTransmitters` class method), 70
`default_type_uid()` (`geoh5py.objects.surveys.magnetics.entirety.EntityType` class method), 71
`default_units` (`geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey` property), 67
`default_units` (`geoh5py.objects.surveys.electromagnetics.grid2d.Grid2D` property), 68
`default_units` (`geoh5py.objects.surveys.electromagnetics.magnetotellurics.MTReceivers` property), 69
`default_units` (`geoh5py.objects.surveys.electromagnetics.magnetotellurics.MTTransmitters` property), 69
`default_vertices` (`geoh5py.objects.geo_image.GeoImage` property), 76
`delete_index_data()` (`geoh5py.shared.concatenation.Concatenator` method), 86
`dependency_requires_value()` (in module `geoh5py.ui_json.utils`), 103
DEPTH (`geoh5py.data.data_association_enum.DataAssociationEnum` attribute), 47
depths (`geoh5py.objects.drillhole.Drillhole` property), 75
description (`geoh5py.shared.entity_type.EntityType` property), 90
desurvey() (`geoh5py.objects.drillhole.Drillhole` method), 75
deviation_x() (in module `geoh5py.objects.drillhole`), 76
deviation_y() (in module `geoh5py.objects.drillhole`), 76
deviation_z() (in module `geoh5py.objects.drillhole`), 76
dict_mapper() (in module `geoh5py.shared.utils`), 92
dip (`geoh5py.objects.grid2d.Grid2D` property), 78
distance_unit (`geoh5py.workspace.workspace.Workspace` property), 107
DrapeModel (class in `geoh5py.objects.drape_model`), 73
Drillhole (class in `geoh5py.objects.drillhole`), 74
DrillholeGroup (class in `geoh5py.groups.drillhole_group`), 52

E

EarthModelsTheme (class in `geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey`), 67
edit_metadata() (`geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey` method), 67
end_of_hole (`geoh5py.objects.drillhole.Drillhole` property), 74
Entity (class in `geoh5py.shared.entity`), 87
entity_type_id (`geoh5py.shared.utils`), 92
entity_type (`geoh5py.data.data.Data` property), 46
entity_type_id (`geoh5py.shared.group.Group` property), 53
entity_type (`geoh5py.objects.object_base.ObjectBase` property), 81
EntityType (`geoh5py.shared.entity.Entity` property), 88
Entirety (class in `geoh5py.shared.entity_type`), 90
extent (`geoh5py.groups.group.Group` property), 53
external_id (`geoh5py.objects.object_base.ObjectBase` property), 81
F
FACE (`geoh5py.data.data_association_enum.DataAssociationEnum` attribute), 47
faces (`geoh5py.objects.object_base.ObjectBase` property), 81
fetch_array_attribute() (`geoh5py.io.h5_reader.H5Reader` class method), 57
fetch_array_attribute() (`geoh5py.workspace.workspace.Workspace` method), 107
fetch_attributes() (`geoh5py.io.h5_reader.H5Reader` class method), 57
fetch_children() (`geoh5py.io.h5_reader.H5Reader` class method), 58
fetch_children() (`geoh5py.workspace.workspace.Workspace` method), 107
fetch_concatenated_attributes() (`geoh5py.io.h5_reader.H5Reader` class method), 58
fetch_concatenated_attributes() (`geoh5py.workspace.workspace.Workspace` method), 108
fetch_concatenated_data_index() (`geoh5py.shared.concatenation.Concatenator` method), 86
fetch_concatenated_list() (`geoh5py.workspace.workspace.Workspace` method), 108
fetch_concatenated_objects() (`geoh5py.shared.concatenation.Concatenator` method), 86
fetch_concatenated_values() (`geoh5py.io.h5_reader.H5Reader` class method), 58

fetch_concatenated_values() (<i>geoh5py.workspace.workspace.Workspace</i> method), 108	fetch_file_object() (<i>geoh5py.io.h5_reader.H5Reader</i> class method), 59	fetch_file_object() (<i>geoh5py.workspace.workspace.Workspace</i> method), 108	fetch_h5_handle() (in module <i>geoh5py.shared.utils</i>), 92	fetch_handle() (<i>geoh5py.io.h5_writer.H5Writer</i> class method), 61	fetch_index() (<i>geoh5py.shared.concatenation.Concatenator</i> method), 86	fetch_metadata() (<i>geoh5py.io.h5_reader.H5Reader</i> class method), 59	fetch_metadata() (<i>geoh5py.objects.surveys.electromagnetic_survey_data.ElectromagneticSurveyData</i> method), 65	fetch_metadata() (<i>geoh5py.workspace.workspace.Workspace</i> method), 108	fetch_or_create_root() (<i>geoh5py.workspace.workspace.Workspace</i> method), 108	fetch_project_attributes() (<i>geoh5py.io.h5_reader.H5Reader</i> class method), 59	fetch_property_groups() (<i>geoh5py.io.h5_reader.H5Reader</i> class method), 59	fetch_property_groups() (<i>geoh5py.workspace.workspace.Workspace</i> method), 108	fetch_start_index() (<i>geoh5py.shared.concatenation.Concatenator</i> method), 86	fetch_type() (<i>geoh5py.io.h5_reader.H5Reader</i> class method), 59	fetch_type() (<i>geoh5py.workspace.workspace.Workspace</i> method), 109	fetch_type_attributes() (<i>geoh5py.io.h5_reader.H5Reader</i> class method), 60	fetch_uuids() (<i>geoh5py.io.h5_reader.H5Reader</i> class method), 60	fetch_value_map() (<i>geoh5py.io.h5_reader.H5Reader</i> class method), 60	fetch_values() (<i>geoh5py.io.h5_reader.H5Reader</i> class method), 60	fetch_values() (<i>geoh5py.shared.concatenation.Concatenator</i> method), 87	fetch_values() (<i>geoh5py.workspace.workspace.Workspace</i> method), 109	file_name (<i>geoh5py.data.filename_data.FileNameData</i> property), 49	file_parameter() (in module <i>geoh5py.ui_json.templates</i>), 100	FILENAME (<i>geoh5py.data.primitive_type_enum.PrimitiveTypeEnum</i> attribute), 50	FilenameData (class in <i>geoh5py.data.filename_data</i>), 49	finalize() (<i>geoh5py.workspace.workspace.Workspace</i> method), 109	find() (<i>geoh5py.shared.entity_type.EntityType</i> class method), 90	find_all() (in module <i>geoh5py.ui_json.utils</i>), 103	find_data() (<i>geoh5py.workspace.workspace.Workspace</i> method), 109	find_entity() (<i>geoh5py.workspace.workspace.Workspace</i> method), 109	find_group() (<i>geoh5py.workspace.workspace.Workspace</i> method), 109	find_object() (<i>geoh5py.workspace.workspace.Workspace</i> method), 109	find_or_create() (<i>geoh5py.data.data_type.DataType</i> class method), 48	find_or_create() (<i>geoh5py.groups.group_type.GroupType</i> class method), 54	find_or_create() (<i>geoh5py.objects.object_type.ObjectType</i> class method), 82	find_or_create_property_group() (<i>geoh5py.objects.object_base.ObjectBase</i> method), 81	find_or_create_property_group() (<i>geoh5py.shared.concatenation.ConcatenatedObject</i> method), 85	find_or_create_type() (<i>geoh5py.groups.group.Group</i> class method), 53	find_or_create_type() (<i>geoh5py.objects.object_base.ObjectBase</i> class method), 81	find_type() (<i>geoh5py.workspace.workspace.Workspace</i> method), 109	fix_up_name() (<i>geoh5py.shared.entity.Entity</i> class method), 88	flatten() (in module <i>geoh5py.ui_json.utils</i>), 103	FLOAT (<i>geoh5py.data.primitive_type_enum.PrimitiveTypeEnum</i> attribute), 50	float_parameter() (in module <i>geoh5py.ui_json.templates</i>), 100	FloatData (class in <i>geoh5py.data.float_data</i>), 49	for_x_data() (<i>geoh5py.data.data_type.DataType</i> class method), 48	for_y_data() (<i>geoh5py.data.data_type.DataType</i> class method), 48	for_z_data() (<i>geoh5py.data.data_type.DataType</i> class method), 48	format_type_string()
---	--	---	---	--	---	--	--	---	--	---	--	---	--	--	---	--	---	---	--	--	---	---	--	--	--	---	--	---	--	--	---	--	--	--	---	---	--	---	---	--	--	--	---	---	--	--	--	--	----------------------

(geoh5py.io.h5_reader.H5Reader static method), 60
 from_ (geoh5py.objects.drillhole.Drillhole property), 75
 from_ (geoh5py.shared.concatenation.ConcatenatedDrillhole property), 84
 from_ (geoh5py.shared.concatenation.ConcatenatedPropertyGroup property), 85

G

ga_version (geoh5py.workspace.workspace.Workspace property), 109
 GeochemistryMineralogyDataSet (class in geoh5py.groups.integrator_group), 54
 GeochemistryMineralogyTheme (class in geoh5py.groups.integrator_group), 54
 geoh5 (geoh5py.workspace.workspace.Workspace property), 109
 Geoh5FileClosedError, 91
 geoh5py
 module, 111
 geoh5py.data
 module, 52
 geoh5py.data.blob_data
 module, 46
 geoh5py.data.color_map
 module, 46
 geoh5py.data.data
 module, 46
 geoh5py.data.data_association_enum
 module, 47
 geoh5py.data.data_type
 module, 47
 geoh5py.data.data_unit
 module, 48
 geoh5py.data.datetime_data
 module, 49
 geoh5py.data.filename_data
 module, 49
 geoh5py.data.float_data
 module, 49
 geoh5py.data.geometric_data_constants
 module, 49
 geoh5py.data.integer_data
 module, 50
 geoh5py.data.numeric_data
 module, 50
 geoh5py.data.primitive_type_enum
 module, 50
 geoh5py.data.reference_value_map
 module, 51
 geoh5py.data.referenced_data
 module, 51
 geoh5py.data.text_data
 module, 51
 geoh5py.data.unknown_data
 module, 52
 geoh5py.groups
 module, 57
 geoh5py.groups.container_group
 module, 52
 geoh5py.groups.custom_group
 module, 52
 geoh5py.groups.drillhole_group
 module, 52
 geoh5py.groups.giftools_group
 module, 53
 geoh5py.groups.group
 module, 53
 geoh5py.groups.group_type
 module, 54
 geoh5py.groups.integrator_group
 module, 54
 geoh5py.groups.maps_group
 module, 56
 geoh5py.groups.notype_group
 module, 56
 geoh5py.groups.property_group
 module, 56
 geoh5py.groups.root_group
 module, 57
 geoh5py.groups.simepeg_group
 module, 57
 geoh5py.groups.survey_group
 module, 57
 geoh5py.io
 module, 64
 geoh5py.io.h5_reader
 module, 57
 geoh5py.io.h5_writer
 module, 60
 geoh5py.objects
 module, 84
 geoh5py.objects.block_model
 module, 72
 geoh5py.objects.curve
 module, 73
 geoh5py.objects.drape_model
 module, 73
 geoh5py.objects.drillhole
 module, 74
 geoh5py.objects.geo_image
 module, 76
 geoh5py.objects.grid2d
 module, 77
 geoh5py.objects.integrator
 module, 79
 geoh5py.objects.label
 module, 79

geoh5py.objects.notype_object
 module, 79
 geoh5py.objects.object_base
 module, 79
 geoh5py.objects.object_type
 module, 82
 geoh5py.objects.octree
 module, 82
 geoh5py.objects.points
 module, 83
 geoh5py.objects.surface
 module, 84
 geoh5py.objects.surveys
 module, 72
 geoh5py.objects.surveys.direct_current
 module, 70
 geoh5py.objects.surveys.electromagnetics
 module, 70
 geoh5py.objects.surveys.electromagnetics.airborne_tem
 module, 64
 geoh5py.objects.surveys.electromagnetics.base
 module, 66
 geoh5py.objects.surveys.electromagnetics.magnet
 module, 68
 geoh5py.objects.surveys.electromagnetics.tipper
 module, 69
 geoh5py.objects.surveys.magnetics
 module, 71
 geoh5py.shared
 module, 97
 geoh5py.shared.concatenation
 module, 84
 geoh5py.shared.entity
 module, 87
 geoh5py.shared.entity_type
 module, 90
 geoh5py.shared.exceptions
 module, 90
 geoh5py.shared.utils
 module, 92
 geoh5py.shared.validators
 module, 94
 geoh5py.shared.weakref_utils
 module, 96
 geoh5py.ui_json
 module, 105
 geoh5py.ui_json.constants
 module, 97
 geoh5py.ui_json.input_file
 module, 97
 geoh5py.ui_json.templates
 module, 99
 geoh5py.ui_json.utils
 module, 103
 geoh5py.ui_json.validation
 module, 105
 geoh5py.workspace
 module, 111
 geoh5py.workspace.workspace
 module, 106
 GeoImage (class in *geoh5py.objects.geo_image*), 76
 GEOMETRIC (*geoh5py.data.primitive_type_enum.PrimitiveTypeEnum*
 attribute), 50
 GeometricDataConstants (class in
 geoh5py.data.geometric_data_constants),
 49
 GeophysicsTheme (class in
 geoh5py.groups.integrator_group), 54
 georeference() (*geoh5py.objects.geo_image.GeoImage*
 method), 76
 georeferencing_from_tiff()
 (*geoh5py.objects.geo_image.GeoImage*
 method), 77
 get_attributes() (in module *geoh5py.shared.utils*), 93
 get_clean_ref() (in module
 geoh5py.shared.weakref_utils), 96
 get_concatenated_attributes()
 (*geoh5py.shared.concatenation.Concatenator*
 method), 87
 get_data() (*geoh5py.objects.object_base.ObjectBase*
 method), 81
 get_data() (*geoh5py.shared.concatenation.ConcatenatedObject*
 method), 85
 get_data_list() (*geoh5py.objects.object_base.ObjectBase*
 method), 81
 get_data_list() (*geoh5py.shared.concatenation.ConcatenatedObject*
 method), 85
 get_entity() (*geoh5py.shared.entity.Entity* method),
 88
 get_entity() (*geoh5py.workspace.workspace.Workspace*
 method), 109
 get_entity_list() (*geoh5py.shared.entity.Entity*
 method), 88
 GifttoolsGroup (class in
 geoh5py.groups.gifttools_group), 53
 Grid2D (class in *geoh5py.objects.grid2d*), 77
 GroundTheme (class in
 geoh5py.groups.integrator_group), 55
 Group (class in *geoh5py.groups.group*), 53
 GROUP (*geoh5py.data.data_association_enum.DataAssociationEnum*
 attribute), 47
 group_enabled() (in module *geoh5py.ui_json.utils*),
 103
 group_optional() (in module *geoh5py.ui_json.utils*),
 103
 group_requires_value() (in module
 geoh5py.ui_json.utils), 103
 groups (*geoh5py.workspace.workspace.Workspace* prop-

erty), 109
 GroupType (class in geoh5py.groups.group_type), 54

H

h5file (geoh5py.workspace.workspace.Workspace property), 110
 H5Reader (class in geoh5py.io.h5_reader), 57
 H5Writer (class in geoh5py.io.h5_writer), 60
 hidden (geoh5py.data.data_type.DataType property), 48

I

image (geoh5py.objects.geo_image.GeoImage property), 77
 image_data (geoh5py.objects.geo_image.GeoImage property), 77
 image_georeferenced (geoh5py.objects.geo_image.GeoImage property), 77
 index (geoh5py.shared.concatenation.Concatenator property), 87
 inf2str() (in module geoh5py.ui_json.utils), 103
 infer_validations() (geoh5py.ui_json.validation.InputValidation static method), 105
 inline_offset (geoh5py.objects.surveys.electromagnetics.airborne_tem.BaseEMSurvey property), 65
 input_type (geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey property), 67
 InputFile (class in geoh5py.ui_json.input_file), 97
 InputValidation (class in geoh5py.ui_json.validation), 105
 insert_once() (in module geoh5py.shared.weakref_utils), 96
 INTEGER (geoh5py.data.primitive_type_enum.PrimitiveType attribute), 50
 integer_parameter() (in module geoh5py.ui_json.templates), 101
 IntegerData (class in geoh5py.data.integer_data), 50
 IntegratorDrillholeGroup (class in geoh5py.groups.drillhole_group), 52
 IntegratorGroup (class in geoh5py.groups.integrator_group), 55
 IntegratorPoints (class in geoh5py.objects.integrator), 79
 IntegratorProject (class in geoh5py.groups.integrator_group), 55
 INVALID (geoh5py.data.primitive_type_enum.PrimitiveType attribute), 50
 is_form() (in module geoh5py.ui_json.utils), 103
 is_uijson() (in module geoh5py.ui_json.utils), 103
 is_uuid() (in module geoh5py.shared.utils), 93
 iterable() (in module geoh5py.shared.utils), 93
 iterable_message() (in module geoh5py.shared.utils), 93
 JSONParameterValidationError, 91
 L
 Label (class in geoh5py.objects.label), 79
 last_focus (geoh5py.objects.object_base.ObjectBase property), 81
 layers (geoh5py.objects.drape_model.DrapeModel property), 73
 list2str() (in module geoh5py.ui_json.utils), 103
 list_data_name (geoh5py.workspace.workspace.Workspace property), 110
 list_entities_name (geoh5py.workspace.workspace.Workspace property), 110
 list_groups_name (geoh5py.workspace.workspace.Workspace property), 110
 list_objects_name (geoh5py.workspace.workspace.Workspace property), 110
 load() (geoh5py.ui_json.input_file.InputFile method), 97
 load_entity() (geoh5py.workspace.workspace.Workspace method), 110
 locations (geoh5py.objects.drillhole.Drillhole property), 75
 loop_radius (geoh5py.objects.surveys.electromagnetics.airborne_tem.BaseEMSurvey property), 65
 M
 map (geoh5py.data.reference_value_map.ReferenceValueMap property), 51
 mapping (geoh5py.data.data_type.DataType property), 48
 MapsGroup (class in geoh5py.groups.maps_group), 56
 mask_by_extent() (in module geoh5py.shared.utils), 93
 match_values() (in module geoh5py.shared.utils), 93
 merge_arrays() (in module geoh5py.shared.utils), 93
 message() (geoh5py.shared.exceptions.AssociationValidationError static method), 90
 message() (geoh5py.shared.exceptions.AtLeastOneValidationError static method), 90
 message() (geoh5py.shared.exceptions.BaseValidationError static method), 91
 message() (geoh5py.shared.exceptions.JSONParameterValidationError static method), 91
 message() (geoh5py.shared.exceptions.OptionalValidationError static method), 91
 message() (geoh5py.shared.exceptions.PropertyGroupValidationError static method), 91
 message() (geoh5py.shared.exceptions.RequiredValidationError static method), 91
 message() (geoh5py.shared.exceptions.ShapeValidationError static method), 91
 message() (geoh5py.shared.exceptions.TypeValidationError static method), 91

`message()` (*geoh5py.shared.exceptions.UUIDValidationError* static method), 92
`message()` (*geoh5py.shared.exceptions.ValueValidationError* static method), 92
`metadata` (*geoh5py.objects.surveys.direct_current.PotentialElectrode* property), 71
`metadata` (*geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey* property), 67
`metadata` (*geoh5py.shared.entity.Entity* property), 89
`modifiable` (*geoh5py.data.data.Data* property), 46
`module`
 geoh5py, 111
 geoh5py.data, 52
 geoh5py.data.blob_data, 46
 geoh5py.data.color_map, 46
 geoh5py.data.data, 46
 geoh5py.data.data_association_enum, 47
 geoh5py.data.data_type, 47
 geoh5py.data.data_unit, 48
 geoh5py.data.datetime_data, 49
 geoh5py.data.filename_data, 49
 geoh5py.data.float_data, 49
 geoh5py.data.geometric_data_constants, 49
 geoh5py.data.integer_data, 50
 geoh5py.data.numeric_data, 50
 geoh5py.data.primitive_type_enum, 50
 geoh5py.data.reference_value_map, 51
 geoh5py.data.referenced_data, 51
 geoh5py.data.text_data, 51
 geoh5py.data.unknown_data, 52
 geoh5py.groups, 57
 geoh5py.groups.container_group, 52
 geoh5py.groups.custom_group, 52
 geoh5py.groups.drillhole_group, 52
 geoh5py.groups.gifttools_group, 53
 geoh5py.groups.group, 53
 geoh5py.groups.group_type, 54
 geoh5py.groups.integrator_group, 54
 geoh5py.groups.maps_group, 56
 geoh5py.groups.notype_group, 56
 geoh5py.groups.property_group, 56
 geoh5py.groups.root_group, 57
 geoh5py.groups.simppeg_group, 57
 geoh5py.groups.survey_group, 57
 geoh5py.io, 64
 geoh5py.io.h5_reader, 57
 geoh5py.io.h5_writer, 60
 geoh5py.objects, 84
 geoh5py.objects.block_model, 72
 geoh5py.objects.curve, 73
 geoh5py.objects.drape_model, 73
 geoh5py.objects.drillhole, 74
 geoh5py.objects.geo_image, 76
 geoh5py.objects.grid2d, 77
 geoh5py.objects.integrator, 79
 geoh5py.objects.label, 79
 geoh5py.objects.notype_object, 79
 geoh5py.objects.object_base, 79
 geoh5py.objects.object_type, 82
 geoh5py.objects.octree, 82
 geoh5py.objects.points, 83
 geoh5py.objects.surface, 84
 geoh5py.objects.surveys, 72
 geoh5py.objects.surveys.direct_current, 70
 geoh5py.objects.surveys.electromagnetics, 70
 geoh5py.objects.surveys.electromagnetics.airborne_tem, 64
 geoh5py.objects.surveys.electromagnetics.base, 66
 geoh5py.objects.surveys.electromagnetics.magnetotelluric, 68
 geoh5py.objects.surveys.electromagnetics.tipper, 69
 geoh5py.objects.surveys.magnetics, 71
 geoh5py.shared, 97
 geoh5py.shared.concatenation, 84
 geoh5py.shared.entity, 87
 geoh5py.shared.entity_type, 90
 geoh5py.shared.exceptions, 90
 geoh5py.shared.utils, 92
 geoh5py.shared.validators, 94
 geoh5py.shared.weakref_utils, 96
 geoh5py.ui_json, 105
 geoh5py.ui_json.constants, 97
 geoh5py.ui_json.input_file, 97
 geoh5py.ui_json.templates, 99
 geoh5py.ui_json.utils, 103
 geoh5py.ui_json.validation, 105
 geoh5py.workspace, 111
 geoh5py.workspace.workspace, 106
`monitored_directory_copy()` (in *module geoh5py.ui_json.utils*), 103
`MTReivers` (class in *geoh5py.objects.surveys.electromagnetics.magnetotellurics*), 68
`MULTI_TEXT` (*geoh5py.data.primitive_type_enum.PrimitiveTypeEnum* attribute), 51
`MultiTextData` (class in *geoh5py.data.text_data*), 51

N

`n_cells` (*geoh5py.objects.block_model.BlockModel* property), 72
`n_cells` (*geoh5py.objects.drape_model.DrapeModel* property), 74
`n_cells` (*geoh5py.objects.grid2d.Grid2D* property), 78

- [n_cells \(geoh5py.objects.object_base.ObjectBase property\), 81](#)
[n_cells \(geoh5py.objects.octree.Octree property\), 83](#)
[n_values \(geoh5py.data.data.Data property\), 46](#)
[n_vertices \(geoh5py.objects.object_base.ObjectBase property\), 81](#)
[name \(geoh5py.data.color_map.ColorMap property\), 46](#)
[name \(geoh5py.data.data_unit.DataUnit property\), 48](#)
[name \(geoh5py.groups.property_group.PropertyGroup property\), 56](#)
[name \(geoh5py.shared.entity.Entity property\), 89](#)
[name \(geoh5py.shared.entity_type.EntityType property\), 90](#)
[name \(geoh5py.ui_json.input_file.InputFile property\), 97](#)
[name \(geoh5py.workspace.workspace.Workspace property\), 110](#)
[ndv \(geoh5py.data.float_data.FloatData property\), 49](#)
[ndv \(geoh5py.data.integer_data.IntegerData property\), 50](#)
[ndv \(geoh5py.data.numeric_data.NumericData property\), 50](#)
[NeighbourhoodSurface \(class in geoh5py.objects.integrator\), 79](#)
[none2str\(\) \(in module geoh5py.ui_json.utils\), 104](#)
[NoTypeGroup \(class in geoh5py.groups.notype_group\), 56](#)
[NoTypeObject \(class in geoh5py.objects.notype_object\), 79](#)
[number_of_bins \(geoh5py.data.data_type.DataType property\), 48](#)
[NumericData \(class in geoh5py.data.numeric_data\), 50](#)
[numify\(\) \(geoh5py.ui_json.input_file.InputFile class method\), 98](#)
- ## O
- [OBJECT \(geoh5py.data.data_association_enum.DataAssociationEnum attribute\), 47](#)
[object_parameter\(\) \(in module geoh5py.ui_json.templates\), 101](#)
[ObjectBase \(class in geoh5py.objects.object_base\), 79](#)
[objects \(geoh5py.workspace.workspace.Workspace property\), 110](#)
[ObjectType \(class in geoh5py.objects.object_type\), 82](#)
[ObservationPointsTheme \(class in geoh5py.groups.integrator_group\), 55](#)
[Octree \(class in geoh5py.objects.octree\), 82](#)
[octree_cells \(geoh5py.objects.octree.Octree property\), 83](#)
[on_file \(geoh5py.shared.entity.Entity property\), 89](#)
[on_file \(geoh5py.shared.entity_type.EntityType property\), 90](#)
[open\(\) \(geoh5py.workspace.workspace.Workspace method\), 110](#)
[optional_parameter\(\) \(in module geoh5py.ui_json.templates\), 102](#)
[optional_requires_value\(\) \(in module geoh5py.ui_json.utils\), 104](#)
[OptionalValidationError, 91](#)
[OptionalValidator \(class in geoh5py.shared.validators\), 94](#)
[options \(geoh5py.groups.simpeg_group.SimPEGGroup property\), 57](#)
[origin \(geoh5py.objects.block_model.BlockModel property\), 72](#)
[origin \(geoh5py.objects.grid2d.Grid2D property\), 78](#)
[origin \(geoh5py.objects.octree.Octree property\), 83](#)
[overwrite_kwargs\(\) \(in module geoh5py.shared.utils\), 94](#)
- ## P
- [parent \(geoh5py.data.color_map.ColorMap property\), 46](#)
[parent \(geoh5py.groups.property_group.PropertyGroup property\), 56](#)
[parent \(geoh5py.groups.root_group.RootGroup property\), 57](#)
[parent \(geoh5py.shared.concatenation.ConcatenatedData property\), 84](#)
[parent \(geoh5py.shared.concatenation.ConcatenatedObject property\), 85](#)
[parent \(geoh5py.shared.concatenation.ConcatenatedPropertyGroup property\), 85](#)
[parent \(geoh5py.shared.entity.Entity property\), 89](#)
[partially_hidden \(geoh5py.shared.entity.Entity property\), 89](#)
[parts \(geoh5py.objects.curve.Curve property\), 73](#)
[path \(geoh5py.ui_json.input_file.InputFile property\), 98](#)
[path2workspace\(\) \(in module geoh5py.ui_json.utils\), 104](#)
[path_name \(geoh5py.ui_json.input_file.InputFile property\), 98](#)
[pitch \(geoh5py.objects.surveys.electromagnetics.airborne_tem.BaseAirborneTem property\), 65](#)
[planning \(geoh5py.objects.drillhole.Drillhole property\), 75](#)
[Points \(class in geoh5py.objects.points\), 83](#)
[potential_electrodes \(geoh5py.objects.surveys.direct_current.CurrentElectrode property\), 71](#)
[potential_electrodes \(geoh5py.objects.surveys.direct_current.PotentialElectrode property\), 71](#)
[PotentialElectrode \(class in geoh5py.objects.surveys.direct_current\), 71](#)
[primitive_type \(geoh5py.data.data_type.DataType property\), 48](#)

[primitive_type\(\)](#) ([geoh5py.data.blob_data.BlobData](#) class method), 46
[primitive_type\(\)](#) ([geoh5py.data.data.Data](#) class method), 47
[primitive_type\(\)](#) ([geoh5py.data.datetime_data.DatetimeData](#) class method), 49
[primitive_type\(\)](#) ([geoh5py.data.filename_data.FilenameData](#) class method), 49
[primitive_type\(\)](#) ([geoh5py.data.float_data.FloatData](#) class method), 49
[primitive_type\(\)](#) ([geoh5py.data.geometric_data_constants.GeometricDataConstants](#) class method), 49
[primitive_type\(\)](#) ([geoh5py.data.integer_data.IntegerData](#) class method), 50
[primitive_type\(\)](#) ([geoh5py.data.numeric_data.NumericData](#) class method), 50
[primitive_type\(\)](#) ([geoh5py.data.referenced_data.ReferencedData](#) class method), 51
[primitive_type\(\)](#) ([geoh5py.data.text_data.CommentsData](#) class method), 51
[primitive_type\(\)](#) ([geoh5py.data.text_data.MultiTextData](#) class method), 51
[primitive_type\(\)](#) ([geoh5py.data.text_data.TextData](#) class method), 52
[primitive_type\(\)](#) ([geoh5py.data.unknown_data.UnknownData](#) class method), 52
[PrimitiveTypeEnum](#) (class in [geoh5py.data.primitive_type_enum](#)), 50
[prisms](#) ([geoh5py.objects.drape_model.DrapeModel](#) property), 74
[properties](#) ([geoh5py.groups.property_group.PropertyGroup](#) property), 56
[property_group](#) ([geoh5py.shared.concatenation.ConcatenatedData](#) property), 84
[property_group_ids](#) ([geoh5py.shared.concatenation.Concatenator](#) property), 87
[property_group_type](#) ([geoh5py.groups.property_group.PropertyGroup](#) property), 56
[property_groups](#) ([geoh5py.objects.object_base.ObjectBase](#) property), 81
[property_groups](#) ([geoh5py.shared.concatenation.ConcatenatedObject](#) property), 85
[PropertyGroup](#) (class in [geoh5py.groups.property_group](#)), 56
[PropertyGroupValidationError](#), 91
[PropertyGroupValidator](#) (class in [geoh5py.shared.validators](#)), 95
[public](#) ([geoh5py.shared.entity.Entity](#) property), 89

Q

[QueryGroup](#) (class in [geoh5py.groups.integrator_group](#)), 55

R

[read_ui_json\(\)](#) ([geoh5py.ui_json.input_file.InputFile](#) static method), 98
[receivers](#) ([geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey](#) property), 67
[reference_to_uid\(\)](#) ([geoh5py.shared.entity.Entity](#) method), 89
[REFERENCED](#) ([geoh5py.data.primitive_type_enum.PrimitiveTypeEnum](#) attribute), 51
[ReferencedData](#) (class in [geoh5py.data.referenced_data](#)), 51
[ReferenceValueMap](#) (class in [geoh5py.data.reference_value_map](#)), 51
[relative_to_bearing](#) ([geoh5py.objects.surveys.electromagnetics.airborne_tem.BaseAir](#) property), 65
[remove_child\(\)](#) ([geoh5py.io.h5_writer.H5Writer](#) static method), 61
[remove_children\(\)](#) ([geoh5py.shared.entity.Entity](#) method), 89
[remove_children\(\)](#) ([geoh5py.workspace.workspace.Workspace](#) method), 110
[remove_children_values\(\)](#) ([geoh5py.objects.object_base.ObjectBase](#) method), 81
[remove_data_from_group\(\)](#) ([geoh5py.data.data.Data](#) method), 47
[remove_data_from_group\(\)](#) ([geoh5py.shared.entity.Entity](#) method), 89
[remove_entity\(\)](#) ([geoh5py.io.h5_writer.H5Writer](#) static method), 61
[remove_entity\(\)](#) ([geoh5py.shared.concatenation.Concatenator](#) method), 87
[remove_entity\(\)](#) ([geoh5py.workspace.workspace.Workspace](#) method), 110
[remove_none_referents\(\)](#) ([geoh5py.workspace.workspace.Workspace](#) method), 110
[remove_none_referents\(\)](#) (in module [geoh5py.shared.weakref_utils](#)), 97
[remove_property_groups\(\)](#) ([geoh5py.objects.object_base.ObjectBase](#) method), 81
[remove_recursively\(\)](#) ([geoh5py.workspace.workspace.Workspace](#) method), 110
[remove_vertices\(\)](#) ([geoh5py.objects.points.Points](#) method), 84
[repack](#) ([geoh5py.workspace.workspace.Workspace](#) property), 110
[RequiredValidationError](#), 91
[RequiredValidator](#) (class in [geoh5py.shared.validators](#)), 95
[requires_value\(\)](#) (in module [geoh5py.ui_json.utils](#)),

104

RockPropertiesTheme (class in geoh5py.groups.integrator_group), 55

roll (geoh5py.objects.surveys.electromagnetics.airborne_tem_base.AirborneTEM property), 65

root (geoh5py.workspace.workspace.Workspace property), 111

RootGroup (class in geoh5py.groups.root_group), 57

rotation (geoh5py.objects.block_model.BlockModel property), 72

rotation (geoh5py.objects.grid2d.Grid2D property), 78

rotation (geoh5py.objects.octree.Octree property), 83

S

SamplesTheme (class in geoh5py.groups.integrator_group), 55

save() (geoh5py.shared.entity.Entity method), 89

save_as() (geoh5py.objects.geo_image.GeoImage method), 77

save_attribute() (geoh5py.shared.concatenation.ConcatenatedPropertyGroup method), 87

save_entity() (geoh5py.io.h5_writer.H5Writer class method), 61

save_entity() (geoh5py.workspace.workspace.Workspace method), 111

save_entity_type() (geoh5py.workspace.workspace.Workspace method), 111

save_file() (geoh5py.data.filename_data.FilenameData method), 49

set_enabled() (in module geoh5py.ui_json.utils), 104

set_metadata() (geoh5py.objects.surveys.electromagnetics.airborne_tem_base.AirborneTEM method), 66

set_tag_from_vertices() (geoh5py.objects.geo_image.GeoImage method), 77

shape (geoh5py.objects.block_model.BlockModel property), 72

shape (geoh5py.objects.grid2d.Grid2D property), 78

shape (geoh5py.objects.octree.Octree property), 83

ShapeValidationError, 91

ShapeValidator (class in geoh5py.shared.validators), 95

SimPEGGroup (class in geoh5py.groups.simpeg_group), 57

sort_depths() (geoh5py.objects.drillhole.Drillhole method), 75

sort_depths() (geoh5py.shared.concatenation.ConcatenatedDrillhole method), 84

str2inf() (in module geoh5py.ui_json.utils), 104

str2list() (in module geoh5py.ui_json.utils), 104

str2none() (in module geoh5py.ui_json.utils), 104

str2uuid() (in module geoh5py.shared.utils), 94

str_from_type() (geoh5py.workspace.workspace.Workspace static method), 111

str_type (geoh5py.io.h5_writer.H5Writer attribute), 62

string_parameter() (in module geoh5py.ui_json.templates), 102

SurfaceId (class in geoh5py.objects.surface), 84

survey_type (geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey property), 67

surveys (geoh5py.objects.drillhole.Drillhole property), 75

T

tag (geoh5py.objects.geo_image.GeoImage property), 77

TEXT (geoh5py.data.primitive_type_enum.PrimitiveTypeEnum attribute), 51

TextData (class in geoh5py.data.text_data), 52

timing_mark (geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey property), 68

TipperBaseStations (class in geoh5py.objects.surveys.electromagnetics.tipper), 69

TipperReceivers (class in geoh5py.objects.surveys.electromagnetics.tipper), 70

to_ (geoh5py.objects.drillhole.Drillhole property), 75

to_ (geoh5py.shared.concatenation.ConcatenatedDrillhole property), 84

to_ (geoh5py.shared.concatenation.ConcatenatedPropertyGroup property), 85

to_geoimage() (geoh5py.objects.grid2d.Grid2D method), 78

to_grid2d() (geoh5py.objects.geo_image.GeoImage method), 77

trace (geoh5py.objects.drillhole.Drillhole property), 75

trace_depth (geoh5py.objects.drillhole.Drillhole property), 75

transmitters (geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey property), 67

transparent_no_data (geoh5py.data.data_type.DataType property), 48

truth() (in module geoh5py.ui_json.utils), 104

type (geoh5py.objects.surveys.electromagnetics.airborne_tem.AirborneTEM property), 64

type (geoh5py.objects.surveys.electromagnetics.airborne_tem.AirborneTEM property), 65

type (geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey property), 68

type (geoh5py.objects.surveys.electromagnetics.magnetotellurics.MTReceivers property), 69

type (geoh5py.objects.surveys.electromagnetics.tipper.TipperBaseStations property), 70

type (geoh5py.objects.surveys.electromagnetics.tipper.TipperReceivers property), 70

types (geoh5py.workspace.workspace.Workspace property), 111

TypeValidationError, 91

TypeValidator (class in geoh5py.shared.validators), 95

U

u_cell_delimiters (geoh5py.objects.block_model.BlockModel property), 72

u_cell_size (geoh5py.objects.grid2d.Grid2D property), 78

u_cell_size (geoh5py.objects.octree.Octree property), 83

u_cells (geoh5py.objects.block_model.BlockModel property), 72

u_count (geoh5py.objects.grid2d.Grid2D property), 78

u_count (geoh5py.objects.octree.Octree property), 83

ui_json (geoh5py.ui_json.input_file.InputFile property), 98

ui_validation() (geoh5py.ui_json.input_file.InputFile class method), 98

uid (geoh5py.groups.property_group.PropertyGroup property), 56

uid (geoh5py.shared.entity.Entity property), 90

uid (geoh5py.shared.entity_type.EntityType property), 90

unique_parts (geoh5py.objects.curve.Curve property), 73

unit (geoh5py.objects.surveys.electromagnetics.base.BaseEMSurvey property), 68

units (geoh5py.data.data_type.DataType property), 48

UNKNOWN (geoh5py.data.data_association_enum.DataAssociationEnum attribute), 47

UnknownData (class in geoh5py.data.unknown_data), 52

update_array_attribute()
(geoh5py.shared.concatenation.Concatenator method), 87

update_attribute() (geoh5py.workspace.workspace.Workspace method), 111

update_attributes()
(geoh5py.shared.concatenation.Concatenator method), 87

update_concatenated_attributes()
(geoh5py.shared.concatenation.Concatenator method), 87

update_concatenated_field()
(geoh5py.io.h5_writer.H5Writer class method), 62

update_field() (geoh5py.io.h5_writer.H5Writer class method), 62

update_ui_values() (geoh5py.ui_json.input_file.InputFile method), 98

uuid2entity() (in module geoh5py.shared.utils), 94

UUIDValidationError, 92

UUIDValidator (class in geoh5py.shared.validators), 96

V

v_cell_delimiters (geoh5py.objects.block_model.BlockModel

property), 72

v_cell_size (geoh5py.objects.grid2d.Grid2D property), 78

v_cell_size (geoh5py.objects.octree.Octree property), 83

v_cells (geoh5py.objects.block_model.BlockModel property), 72

v_count (geoh5py.objects.grid2d.Grid2D property), 78

v_count (geoh5py.objects.octree.Octree property), 83

validate() (geoh5py.shared.validators.AssociationValidator class method), 94

validate() (geoh5py.shared.validators.AtLeastOneValidator class method), 94

validate() (geoh5py.shared.validators.BaseValidator class method), 94

validate() (geoh5py.shared.validators.OptionalValidator class method), 94

validate() (geoh5py.shared.validators.PropertyGroupValidator class method), 95

validate() (geoh5py.shared.validators.RequiredValidator class method), 95

validate() (geoh5py.shared.validators.ShapeValidator class method), 95

validate() (geoh5py.shared.validators.TypeValidator class method), 95

validate() (geoh5py.shared.validators.UUIDValidator class method), 96

validate() (geoh5py.shared.validators.ValueValidator class method), 96

validate() (geoh5py.ui_json.validation.InputValidation method), 105

validate_data() (geoh5py.objects.drillhole.Drillhole method), 75

validate_data() (geoh5py.shared.concatenation.ConcatenatedDrillhole method), 84

validate_data() (geoh5py.ui_json.validation.InputValidation method), 105

validate_data_association()
(geoh5py.objects.object_base.ObjectBase method), 82

validate_data_type()
(geoh5py.objects.object_base.ObjectBase static method), 82

validate_interval_data()
(geoh5py.objects.drillhole.Drillhole method), 75

validate_interval_data()
(geoh5py.shared.concatenation.ConcatenatedDrillhole method), 85

validate_log_data()
(geoh5py.objects.drillhole.Drillhole method), 75

validation_options (geoh5py.ui_json.input_file.InputFile property), 98

validations (*geoh5py.ui_json.input_file.InputFile* attribute), 47
 property), 98
 validations (*geoh5py.ui_json.validation.InputValidation* property), 105
 validator_type (*geoh5py.shared.validators.AssociationValidator* attribute), 94
 validator_type (*geoh5py.shared.validators.AtLeastOneValidator* attribute), 94
 validator_type (*geoh5py.shared.validators.BaseValidator* property), 94
 validator_type (*geoh5py.shared.validators.OptionalValidator* attribute), 95
 validator_type (*geoh5py.shared.validators.PropertyGroupValidator* attribute), 95
 validator_type (*geoh5py.shared.validators.RequiredValidator* attribute), 95
 validator_type (*geoh5py.shared.validators.ShapeValidator* attribute), 95
 validator_type (*geoh5py.shared.validators.TypeValidator* attribute), 96
 validator_type (*geoh5py.shared.validators.UUIDValidator* attribute), 96
 validator_type (*geoh5py.shared.validators.ValueValidator* attribute), 96
 validators (*geoh5py.ui_json.input_file.InputFile* property), 98
 validators (*geoh5py.ui_json.validation.InputValidation* property), 105
 value_map (*geoh5py.data.data_type.DataType* property), 48
 value_map (*geoh5py.data.referenced_data.ReferencedData* property), 51
 values (*geoh5py.data.color_map.ColorMap* property), 46
 values (*geoh5py.data.data.Data* property), 47
 values (*geoh5py.data.filename_data.FilenameData* property), 49
 values (*geoh5py.data.integer_data.IntegerData* property), 50
 values (*geoh5py.data.numeric_data.NumericData* property), 50
 values (*geoh5py.data.text_data.CommentsData* property), 51
 values (*geoh5py.data.text_data.MultiTextData* property), 52
 values (*geoh5py.data.text_data.TextData* property), 52
 ValueError, 92
 ValueValidator (class in *geoh5py.shared.validators*), 96
 VECTOR (*geoh5py.data.primitive_type_enum.PrimitiveTypeEnum* attribute), 51
 version (*geoh5py.workspace.workspace.Workspace* property), 111
 VERTEX (*geoh5py.data.data_association_enum.DataAssociationEnum* attribute), 47
 vertical (*geoh5py.objects.grid2d.Grid2D* property), 78
 vertical_offset (*geoh5py.objects.surveys.electromagnetics.airborne_tem_surveys.AirborneTEMSurvey* property), 66
 vertices (*geoh5py.objects.geo_image.GeoImage* property), 77
 vertices (*geoh5py.objects.object_base.ObjectBase* property), 82
 vertices (*geoh5py.objects.points.Points* property), 84
 visible (*geoh5py.shared.entity.Entity* property), 90
W
 w_cell_size (*geoh5py.objects.octree.Octree* property), 83
 w_count (*geoh5py.objects.octree.Octree* property), 83
 waveform (*geoh5py.objects.surveys.electromagnetics.base.BaseTEMSurvey* property), 68
 waveform_parameters (*geoh5py.objects.surveys.electromagnetics.base.BaseTEMSurvey* property), 68
 workspace (class in *geoh5py.workspace.workspace*), 106
 workspace (*geoh5py.shared.entity.Entity* property), 90
 workspace (*geoh5py.shared.entity_type.EntityType* property), 90
 workspace (*geoh5py.ui_json.input_file.InputFile* property), 98
 workspace (*geoh5py.ui_json.validation.InputValidation* property), 105
 workspace (*geoh5py.workspace.workspace.Workspace* property), 111
 workspace2path() (in module *geoh5py.ui_json.utils*), 104
 write_array_attribute() (*geoh5py.io.h5_writer.H5Writer* class method), 62
 write_attributes() (*geoh5py.io.h5_writer.H5Writer* class method), 62
 write_color_map() (*geoh5py.io.h5_writer.H5Writer* class method), 62
 write_data_values() (*geoh5py.io.h5_writer.H5Writer* class method), 63
 write_entity() (*geoh5py.io.h5_writer.H5Writer* class method), 63
 write_entity_type() (*geoh5py.io.h5_writer.H5Writer* class method), 63
 write_file_name_data() (*geoh5py.io.h5_writer.H5Writer* class method), 63
 write_properties() (*geoh5py.io.h5_writer.H5Writer* class method), 63
 write_property_groups() (*geoh5py.io.h5_writer.H5Writer* class method), 63

`write_to_parent()` (*geoh5py.io.h5_writer.H5Writer*
class method), [63](#)
`write_ui_json()` (*geoh5py.ui_json.input_file.InputFile*
method), [98](#)
`write_value_map()` (*geoh5py.io.h5_writer.H5Writer*
class method), [64](#)
`write_visible()` (*geoh5py.io.h5_writer.H5Writer*
class method), [64](#)

X

`x_datatype_uid()` (*geoh5py.data.geometric_data_constants.GeometricDataConstants*
class method), [49](#)

Y

`y_datatype_uid()` (*geoh5py.data.geometric_data_constants.GeometricDataConstants*
class method), [49](#)
`yaw` (*geoh5py.objects.surveys.electromagnetics.airborne_tem.BaseAirborneTEM*
property), [66](#)

Z

`z_cell_delimiters` (*geoh5py.objects.block_model.BlockModel*
property), [72](#)
`z_cells` (*geoh5py.objects.block_model.BlockModel*
property), [72](#)
`z_datatype_uid()` (*geoh5py.data.geometric_data_constants.GeometricDataConstants*
class method), [49](#)