

## Table of Contents

Overview .....	2
Requirements .....	2
Capabilities and Workflow options.....	3
Input INI file structure.....	4
Command-Line Usage / Execution.....	13
Command-Line Input Parameters .....	13
Command-Line Examples.....	14
Guidance, Limitations, and Known Issues.....	15
Release History .....	16

## Overview

The Json2GeoJSON.py Python script is a Conversion routine designed to be run by the OverwriteFS.py script. The purpose of this script is to read Json data files and convert them to GeoJSON feature collection files. Geographic data is recorded as GeoJSON Geometries. Content without a Geometry will be assigned a Point location of Latitude and Longitude of 0, 0. If Longitude and Latitude fields are provided, using the 'xField' and 'yField' properties, this script will convert the values to Point features. Multiple Geometry feature types are supported (point, line, and polygon), creating one layer per geometry type. Also supports Multi-Part Geometries per Feature of the same type.

This routine will generate a Text INI file alongside the input file. This INI file can be used to Order and Name the fields that are written to the GeoJSON output file for Overwriting or updating a Service. The INI file also allows for data extraction from the field value before saving, supporting simple search, offset, and length operations along with Concatenation and basic math for field conversions. Example would be to extract properties from Comment or Description text in the data and save as separate fields. Or to convert speed of Mph to Km/h.

This script allows for the detection and comparison of the Last Publication date/time to the current publication based on element entries of 'pubDate', 'published', 'generated' and 'lastBuildDate'. If there has been no change, it returns an empty string to the calling routine (OverwriteFS.py script). This indicates no change to the data and therefore makes no change to the service. This feature can be turned off by including 'False' as the first optional parameter when calling the routine. The most recent Publication date is stored in the INI file's properties section.

## Requirements

1. Python 3.x or Python Notebook
2. Access to dependent script 'datetimeUtils.py' in 'Support' folder
3. A source Json or GeoJSON formatted file.

## Capabilities and Workflow options

- Json 'key: value' pairs make up the source data. Where the 'key' Element names are used as field names. The Element 'value' is used as the field data. See the INI description for options on extracting specific content from an Element's text.
- Fields can be included that have no Element source. The INI field Default value can be used to hydrate the field data, or the field Type default will be assigned.
- Fields can be created from a source Element data by extracting content using the INI field definition. Field data can be extracted, concatenated, or manipulated as needed. Fields can be created using specified text or values, then altered using existing field data.
- The list of Items read from the Json will default to 'features' for GeoJSON. A user can specify a "Root Element" to access if the item list is named differently.
- Geometry construction can originate X, Y, and Z coordinate values to come from field content. The Z coordinate of existing Geometries can be added using a specified Z-Field or altered using an Offset or Product value.

## Input INI file structure

When reading the input data file, this script will generate and maintain a Text based INI file that retains details specific to the input file, like feed properties and field definitions. The INI file will be named after the input filename and will be placed alongside the input file.

The INI file structure follows the typical INI structure yet includes some additions that allow for data extraction and creation of new fields. The structure contains Two 'Key=Value' style properties groups consisting of a set of 'Processing' properties and 'File' Field properties. See following format:

```
[properties]
lastPublicationDate = 2021/09/04 07:40:23
rootElement = <'item' Element Name>
flattenData = True
flattenNames = True
exclude = <Element Name>
exclude = <...>
trimOuterSpaces = True
allowNulls = True
xField = <X or Longitude Field Name>
yField = <Y or Latitude Field Name>
zField = <Z or Elevation Field Name>
zFactor = <Float or Integer Z Multiplier>
zOffset = <Float or Integer Z Incrementor>

[<input filename>.json]
<Element Name> = <output field name> [<output field type> [<optional properties>]]
<Element Name> = <output field name> [<output field type> [<optional properties>]]
<...>
```

- The '***properties***' or 'processing section' contains feed specific processing details that are carried from run to run.
  - o '***lastPublicationDate***' stores the Publication Date of the most recent Json file. This is compared to the next download to determine if there is a change in the publication data.
  - o '***rootElement***' stores the list element name that should be processed to generate the GeoJSON data output. One item in the list per feature output. Default is 'features' for GeoJSON formatted input files.
  - o '***flattenData***' is set as a True or False flag that tells the script to flatten all Sub-Elements (or Dictionary elements) into fields. True turns on flattening and False turns it off. When enabled, the generated field names will include (<element>\_<sub-element>\_<sub-element>\_...) when the '***flattenNames***' property is set to False. Default is True
  - o '***flattenNames***' is set as a True or False flag that tells the script to flatten the field Name for all Sub-Elements to use only the '<sub-element>' name when set to True. See '***flattenData***' property for False setting. Only valid when '***flattenData***' is True. Default is True
  - o '***exclude***' is available when '***flattenData***' is True. This allows you to specify the Element name NOT to flatten. This supports one Element per '***exclude***' property, include as many '***exclude***' properties as needed!

- **'trimOuterSpaces'** is set as a True or False flag that tells the script to trim leading and trailing spaces from the field when reading or extracting data. Default is True
- **'allowNulls'** is set as a True or False flag that tells the script to output Null field values when the data is empty, or the value is the same as the [<output field type>](#) Default value. Default is True
- **'xField'** is set as a the [<output field name>](#) that will contain the Longitude or 'X' coordinate value used to populate the Point Geometry field when no other Geometry value is available. **\* Note \* Only valid for Point Geometries and MUST be an Integer or Float <output field type>!**
- **'yField'** is set as a the [<output field name>](#) that will contain the Latitude or 'Y' coordinate value used to populate the Point Geometry field when no other Geometry value is available. **\* Note \* Only valid for Point Geometries and MUST be an Integer or Float <output field type>!**
- **'zField'** is set as a the [<output field name>](#) that will contain the Depth / Elevation or 'Z' coordinate value used to populate the Point Geometry field when no other Geometry value is available. If the row contains a Geometry, but does not include a Z value, setting this property will add a Z value to the existing Geometry! **\* Note \* Only valid for Point or Multi-Point Geometries and MUST be an Integer or Float <output field type>!**
- **'zFactor'** is set as a float or integer value that can be used to Multiply the 'Z' coordinate value, used to alter, or exaggerate the Point Geometry's elevation or depth. If no Geometry value is available or the Geometry does not include a Z coordinate, no changes will be made! This feature can be used to alter the absolute Depth reading, setting it to a Negative Elevation. Like changing a 14km deep Earthquake into a -14,000m elevation value by multiplying the depth by -1000. Default value is 1. **\* Note \* Only valid for Point or Multi-Point Geometries!**
- **'zOffset'** is set as a float or integer value that can be used to Add or Subtract from the 'Z' coordinate value, used to alter, or exaggerate the Point Geometry's elevation or depth. If no Geometry value is available or the Geometry does not contain a Z coordinate, no changes will be made! Default is 0. **\* Note \* Only valid for Point or Multi-Point Geometries!**

- The '*[<input filename>.json]*' or 'fields section' contains field layout details for the input file.
  - o '*<element name>*' is Required and will contain the Element name identified in the Json data, or the *<output field name>* of any field processed prior to this field (above it in the list of fields) if a matching Element name is not available. The Value for the field will be pulled from this Element's content. Elements that do not exist in the Json data can be defined in the INI file, adding fields to the output. These fields can be schema additions for later computation or stand ins for fields that have not yet been created in the Json. For computed fields, this can be 'NonOp' (non-operational) or some other nondescript name, as it has no real Element source.
  - o '*<output field name>*' is Required and will become the field name in the output file. The field name is initially assigned base on the Element name and if the '*flattenNames*' property is used. In cases where a name already exists, a numbering scheme of 2; 3; 4; and so on will be added to the right side of the name for uniqueness, this applies to both the Element Name and the Field Name. Ex. Multiple 'data\_value' fields become 'data\_value', 'data\_value2', 'data\_value3', and so on. **\* Note \* The maximum number of characters accepted is 31, ArcGIS Online will truncate field names longer than this, with unique names being lost. Field output is disabled.** Default is the same as *<element name>*. A message is displayed if length exceeded!
  - o '*<output field type>*' is Optional, containing the *suggested* field type that should be used in the Service. **\* Note \* This is based on the field contents and is ultimately set by the Publishing logic during the Overwrite.** If the specified type is 'date', the field value will be examined for any Date/Time content using the 'datetimeUtils' routine, it can even detect an Epoch value! Supported types include:
    - '*text*' is a string character array field. **This is the Default field type!**
      - Default value is an empty string: ""
    - '*integer*' is a 32-bit integer number field. Value output is a Json Number!
      - Default value is integer: 0
    - '*float*' is a Double-Precision floating point field. Value output is a Json Number!
      - Default value is float: 0.0
    - '*date*' is a Date/Time field. Value output is a text Date/Time string!
      - Default value is epoch 0, as date string: "1970/01/01T00:00:00"
  - o '*[<optional properties>]*' are of course optional, but if used you must specify an *<output field type>* property first! Entries here are processed as space delimited pairs of space separated '*<setting> <value>*' entries support the following, unless otherwise specified. **\* Note \* that any required spaces in the setting 'value' should be replaced by a URL encoded '%20' for input processing compatibility:**
    - '*Width*' sets the width of text field. Though Publishing minimum for GeoJSON is 256. The actual field width is set by the largest data value encountered during publishing. Setting this value here will *Truncate* values found that are larger than what is specified and the first record will receive a *Space Padded* value to set the desired width, otherwise the published field width will vary from update to update. Ex. 'width 1024'
    - '*Case*' sets the output string case of text field. Supported values are '*Upper*' (all letters are capitalized), '*Lower*' (no letters are capitalized), '*Capital*' (first word in phrase is capitalized), '*AllCapital*' (all words in phrase are capitalized), '*Title*' (all

words in a phrase are capitalized except minor words. Ex 'The state of all things' becomes 'The State of All Things'), '**Camel**' (spaces in a phrase are removed and all words are capitalized. Ex 'The professional group' becomes 'TheProfessionalGroup'), '**camel**' (spaces in a phrase are removed and all words but the first are capitalized. Ex 'I phone' becomes 'iPhone', 'Camel Case' becomes 'camelCase'), and '**Acronym**' (use first letter from all words with existing case. Ex 'Department of Justice' becomes 'DoJ', 'Environmental systems research institute' becomes 'Esri'). Default: no case change is made.

- '**Default**' is a value that can be included as a default value if the source [<element name>](#) does not exist, forcing the publishing action to create a field type other than the default 'text' field. **\* Tip \*** The 'value' entry for this property can also include the '[<output field name>](#)' of any field processed prior to this field (above it in the list of fields). This field's value will be used instead of a constant. If a default is not provided, the [<output field type>](#) default will be used. Ex. 'Default This%20is%20a%20test!'
  - '**AllowNulls**' is a Flag setting that does **NOT** use a 'value', it is a standalone field property! When included, the field value will be saved as 'null' if the field content is the same as the [<output field type>](#) Default value, overriding the 'processing section' setting for this field only!
  - '**DoNotSave**' is a Flag setting that does **NOT** use a 'value', it is a standalone field property! When included, the field will **NOT** be saved to the output. This is handy for building derived or intermediate fields for processing Element data that will be used by other fields and you may not want to include them in the output.
- '[\[<optional properties>\]](#)' Element data Extraction can be handled by using the following optional field property entries. These properties are processed in a linear fashion. Chain these together on the field definition to develop the field processing needed to 'extract' or augment data in the Element's content. **\* Tip \*** The 'value' entry for these properties can also include the '[<output field name>](#)' of any field processed prior to this field (above it in the list of fields). This field's value will be used instead of a constant.
- '**Offset**' will start extraction evaluation at offset position Left of Element content. A negative value starts n characters from the right. Used without any other extraction properties will result in truncating content LEFT of this position! Default is '0'
  - '**Length**' will end extraction at length, or number of characters. Used without any other extraction properties will result in truncating content to the RIGHT of this length! Default is length of Element content remaining.
  - '**Start**' is the beginning search text, starting data extraction at the next character to the right of finding the 'start' text. Used without any other extraction properties will result in truncating content LEFT of and including the 'start' text! Default is no start search.
  - '**End**' is the ending search text to look for, stopping the extraction at the character to the left of the end text. Used without any other extraction properties will result in truncating content to the RIGHT of where this 'end' text begins! Default is length of Element content remaining.
  - '**Concat**' specifies a value or Field data to Concatenate to the right of this field content. Use this to merge static or dynamic field data together.

- **'Add'** specified a numeric value or Field data to add to this field content. Use this to modify the numeric value of the field data.
- **'Sub'** specified a numeric value or Field data to subtract from this field content. Use this to modify the numeric value of the field data.
- **'Mult'** specified a numeric value or Field data to multiply this field content by. Use this to modify the numeric value of the field data. Ex. Use to convert unit value from one type to another, MPH to Km/h for instance.
- **'Div'** specified a numeric value or Field data to divide this field content by. Use this to modify the numeric value of the field data.

As an example, consider the following GeoJSON item data taken from USGS Earthquake site. This is an excerpt taken from the 'feature' [rootElement](#) list containing many items in the Json data.

Initially, the **Highlighted** Elements will be used to create fields in the output GeoJSON, each being added to the output INI file automatically when the field section or list of fields are not defined. The Geometry 'point' Elements in the source data will be used to create the Point GeoJSON Layer.

```
{
  "type": "Feature",
  "properties": {
    "mag": 4.8,
    "place": "72 km E of Hualien City, Taiwan",
    "time": 1636482254486,
    "updated": 1636484113040,
    "tz": null,
    "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000fss1",
    "detail": "https://earthquake.usgs.gov/fdsnws/event/1/query?eventid=us7000fss1&format=geojson",
    "felt": null,
    "cdi": null,
    "mmi": null,
    "alert": null,
    "status": "reviewed",
    "tsunami": 0,
    "sig": 354,
    "net": "us",
    "code": "7000fss1",
    "ids": ",us7000fss1,",
    "sources": ",us,",
    "types": ",origin,phase-data,",
    "nst": null,
    "dmin": 0.679,
    "rms": 0.79,
    "gap": 61,
    "magType": "mb",
    "type": "earthquake",
    "title": "M 4.8 - 72 km E of Hualien City, Taiwan"
  },
  "geometry": {
    "type": "Point",
    "coordinates": [122.3123, 23.9958, 27.65]
  },
  "id": "us7000fss1"
}
```

Here is the initial, or default, INI representation of the above data:

```
[properties]
lastPublicationDate = 2021/11/10 06:02:23
rootElement = features
flattenData = True
flattenNames = True
trimOuterSpaces = True
allowNulls = True
xField =
yField =
zField =
zFactor = 1.0
zOffset = 0.0

[EarthquakeQuery.geojson]
id = id
properties_alert = alert
properties_cdi = cdi
properties_code = code
properties_detail = detail
properties_dmin = dmin
properties_felt = felt
properties_gap = gap
properties_ids = ids
properties_mag = mag
properties_magType = magType
properties_mmi = mmi
properties_net = net
properties_nst = nst
properties_place = place
properties_rms = rms
properties_sig = sig
properties_sources = sources
properties_status = status
properties_time = time
properties_title = title
properties_tsunami = tsunami
properties_type = type2
properties_types = types
properties_tz = tz
properties_updated = updated
properties_url = url
type = type text DoNotSave
```

Customizations can be performed, whereby the output Field names can be changed, the data types can be adjusted, the field order can be rearranged, fields can be added or removed, and ad-hoc fields can be created by extracting content from existing fields. As an example, examine the sample item data in our initial example. You'll find that the 'time' and 'updated' date fields are set to Epoch seconds and the Geometry Z value is an absolute Depth in Kilometers. We can refine the INI file to improve the clarity of these and other fields.

To adjust the Geometry Z values, we can alter the '[zFactor](#)' parameter to change the Kilometer depth to Negative Meters by setting it to '-1000'. This will be used to multiply each Z value, turning them into negative meters. Ex. Z = 27.65, will become -27650

Setting the field types can convert values like dates to be represented as dates, numbers will be represented as numbers. The 'updated' field contains Epoch numbers. Setting this field type to a 'date' type will force the value to be processed as a date, which detects the Epoch value, returning it as a date/time string. Ex. Epoch 'time' value of '1636482254486' will become '2021-11-09 18:24:14' (without milliseconds)

Here's a look at a configured INI file where we have included the specific fields types and exclude other fields that aren't adding value.

You can see the **yellow** highlighted field types we added and the zFactor setting we changed. The **light blue** highlighted fields are set not to be saved. The 'type' Element is automatically set not to save because it is the source GeoJSON 'Feature' type designator, not a true field in the data! The 'tz' or time zone field does not provide value, as it is not being populated.

```
[properties]
lastPublicationDate = 2021/11/10 06:02:23
rootElement = features
flattenData = True
flattenNames = True
trimOuterSpaces = True
allowNulls = True
xField =
yField =
zField =
zFactor = -1000.0
zOffset = 0.0

[EarthquakeQuery.geojson]
id = id
properties_alert = alert
properties_cdi = cdi integer
properties_code = code
properties_detail = detail
properties_dmin = dmin float
properties_felt = felt integer
properties_gap = gap integer
properties_ids = ids
properties_mag = mag float
properties_magType = magType
properties_mmi = mmi float
properties_net = net
properties_nst = nst
properties_place = place
properties_rms = rms float
properties_sig = sig integer
properties_sources = sources
properties_status = status
properties_time = time date
properties_title = title
properties_tsunami = tsunami integer
properties_type = type
properties_types = types
properties_tz = tz text DoNotSave
properties_updated = updated date
properties_url = url
type = type text DoNotSave
```

The resulting GeoJSON data for the above example shows how this alters the output:

```

{
  "type": "Feature",
  "properties": {
    "id": "us7000fssl",
    "alert": "",
    "cdi": 0,
    "code": "7000fssl",
    "detail": "https://earthquake.usgs.gov/fdsnws/event/1/query?eventid=us7000fssl&format=geojson",
    "dmin": 0.679,
    "felt": 0,
    "gap": 61,
    "ids": ",us7000fssl,",
    "mag": 4.8,
    "magType": "m",
    "mmi": 0.0,
    "net": "us",
    "nst": "",
    "place": "72 km E of Hualien City, Taiwan",
    "rms": 0.79,
    "sig": 354,
    "sources": ",us,",
    "status": "reviewed",
    "time": "2021-11-09 18:24:14",
    "title": "M 4.8 - 72 km E of Hualien City, Taiwan",
    "tsunami": 0,
    "type": "earthquake",
    "types": ",origin,phase-data,",
    "updated": "2021-11-09 18:55:13",
    "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us7000fssl",
  },
  "geometry": {
    "type": "Point",
    "coordinates": [122.3123, 23.9958, -27650.0]
  }
}

```

The next example is a Json dataset that comes from a General Transit Feed Specification or [GTFS](#) data source. This data is formatted as an array of dictionaries, each entry in the array is a transit record.

```

[
  {
    "ty": "gtfs",
    "l": {
      "pid": 661,
      "n": "Izmit",
      "t": "Izmit, \u0130zmit/Kocaeli, Turkey",
      "lat": 40.765441,
      "lng": 29.940809,
      "id": 662
    },
    "u": {
      "d": "http://kocaeli.bel.tr/webfiles/userfiles/files/birimler/bilgi-islem-dairesi-baskanligi/kocaeli-gtfs.zip"
    },
    "t": "Kocaeli GTFS",
    "id": "kocaeli-buyuksehir-belediyesi/964",
    "latest": {
      "ts": 1540898966
    }
  }
]

```

For this data, there is no [rootElement](#), as the file contains a single array of entries. This root entry list becomes the source “list” of entries to process. The INI file for this data has been setup to leverage the “lat” field as the ‘yField’ and “lng” field as the ‘xField’ since there is no Geometry value. Like our last example, we have provided field types and converted the date field from an Epoch value to a date. We also clarified some of the field names to make this data more understandable. Here’s a look at the configured INI file for this data:

```

[properties]
lastPublicationDate =
rootElement =
flattenData = True
flattenNames = False
trimOuterSpaces = True
allowNulls = True
xField = lng
yField = lat
zField =
zFactor = 1.0
zOffset = 0.0

[AllFeeds.jsonlist]
id = id
l_id = location_id integer
l_lat = lat float DoNotSave
l_lng = lng float DoNotSave
l_n = location_name
l_pid = location_pid integer
l_t = location_long_name
latest_ts = latest_timestamp date
t = title
ty = data_type
u_d = u_d
u_i = u_i

```

The output GeoJSON data is ready for upload.

```

{
  "type": "Feature",
  "properties": {
    "id": "kocaeli-buyuksehir-belediyesi/964",
    "location_id": 662,
    "location_name": "Izmit",
    "location_pid": 661,
    "location_long_name": "Izmit, \u0130zmit/Kocaeli, Turkey",
    "latest_timestamp": "2018-10-30 11:29:26",
    "title": "Kocaeli GTFS",
    "data_type": "gtfs",
    "u_d": "http://kocaeli.bel.tr/webfiles/userfiles/files/birimler/bilgi-islem-dairesi-
baskanligi/kocaeli-gtfs.zip",
    "u_i": ""
  },
  "geometry": {
    "type": "Point",
    "coordinates": [29.940809, 40.765441]
  }
}

```

## Command-Line Usage / Execution

To execute script, open a Python Command Line Window and type:

```
'Python Converters\Json2GeoJSON.py <sourceFilename> [<checkPublication> [<verbose>]'
```

## Command-Line Input Parameters

- Available Input Parameters

<sourceFilename>: (required) String Path and or Filename of source Json file to process.

<checkPublication>: (optional) Boolean True or False telling script to compare file's Publication Date to the most recently processed Publication Date. If not newer, return an empty string that tells the calling routine (OverwriteFS script) that there is no change to the data.

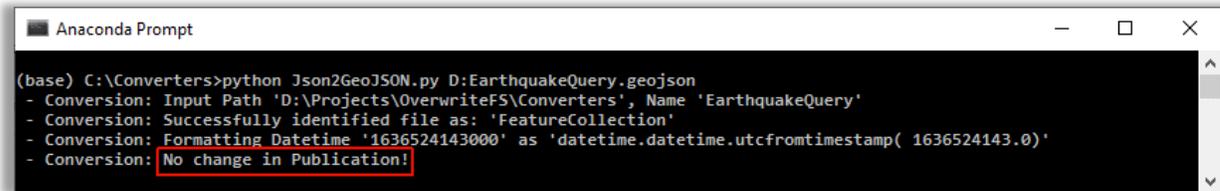
Default is True

<verbose>: (optional) Boolean True or False telling script to display progress and details. Setting this to False will turn off progress reporting altogether!

Default is True

## Command-Line Examples

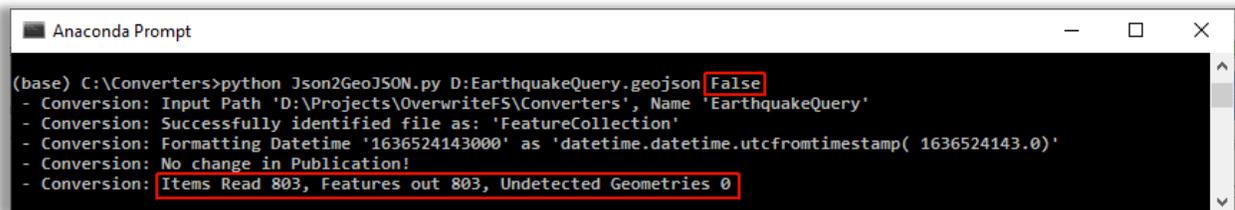
- Execution with just the source Filename. Script shows input file path, name, and detected type. It also shows the evaluated Publication Date/Time format and value, reporting there has been no change to the data since last run.



```

Anaconda Prompt
(base) C:\Converters>python Json2GeoJSON.py D:EarthquakeQuery.geojson
- Conversion: Input Path 'D:\Projects\OverwriteFS\Converters', Name 'EarthquakeQuery'
- Conversion: Successfully identified file as: 'FeatureCollection'
- Conversion: Formatting Datetime '1636524143000' as 'datetime.datetime.utcnow().strftime('%Y%m%d%H%M%S.%f')'
- Conversion: No change in Publication!
  
```

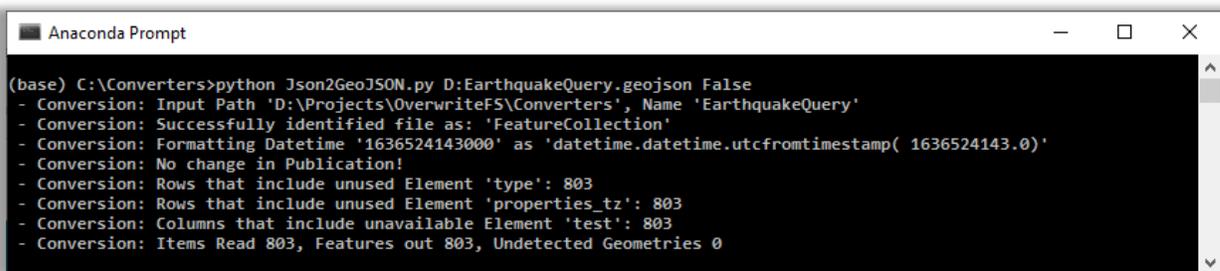
- Example run that overrides the 'checkPublication' value, set to False and ignoring the PubDate. This forces the routine to process the data even if there has been no change. The additional detail reports the number of rows read in and written out.



```

Anaconda Prompt
(base) C:\Converters>python Json2GeoJSON.py D:EarthquakeQuery.geojson False
- Conversion: Input Path 'D:\Projects\OverwriteFS\Converters', Name 'EarthquakeQuery'
- Conversion: Successfully identified file as: 'FeatureCollection'
- Conversion: Formatting Datetime '1636524143000' as 'datetime.datetime.utcnow().strftime('%Y%m%d%H%M%S.%f')'
- Conversion: No change in Publication!
- Conversion: Items Read 803, Features out 803, Undetected Geometries 0
  
```

- This script will display a column count for any field defined where the Element specified is NOT found in the source data, typical for ad-hoc fields added by the user. Here, you can see a 'test' field that was added, pointing to a test Element that does not exist. It also displays a row count for the number of rows in the source data that have a specific Element that is not used for output. This script will also report any field that has trouble being processed.



```

Anaconda Prompt
(base) C:\Converters>python Json2GeoJSON.py D:EarthquakeQuery.geojson False test
- Conversion: Input Path 'D:\Projects\OverwriteFS\Converters', Name 'EarthquakeQuery'
- Conversion: Successfully identified file as: 'FeatureCollection'
- Conversion: Formatting Datetime '1636524143000' as 'datetime.datetime.utcnow().strftime('%Y%m%d%H%M%S.%f')'
- Conversion: No change in Publication!
- Conversion: Rows that include unused Element 'type': 803
- Conversion: Rows that include unused Element 'properties_tz': 803
- Conversion: Columns that include unavailable Element 'test': 803
- Conversion: Items Read 803, Features out 803, Undetected Geometries 0
  
```

## Guidance, Limitations, and Known Issues

- **Guidance**
  - Always a good idea to start a new Service by downloading your data file and running this script manually. This allows you to prototype the design without adding Service maintenance into the mix until you have a design you are happy with.
  - When you complete your design changes and applied your updates, make a copy of the INI file for your records. Losing this file would force you to start your design work over gain from scratch!
  - Once the 'field section' of the INI file contains entries, the script will NOT add new fields, as not to corrupt the existing schema. Yet they can be added manually!
  
- **Limitations**
  - Detailed and precise schema control is not available in this release. Future releases will hopefully provide this capability. May need to redesign the underlying calls to be able to control the schema passed to the Portal REST endpoints.
  
- **Known Issues**
  - When running parent OverwriteFS script through an online Notebook, include the 'outPath' option, specifying a folder within the '/arcgis/home' directory. If not, the default temp location could drop your download and INI files during cleanup, making the service schema inconsistent. Same is true when storing files outside of the 'home' folder, these are subject to removal by the Notebook Kernel. The temp location may also be difficult to access, making alterations to the INI file near impossible!

## Release History

- **November 2021, v1.0.0:** Initial public release.
- **December 2021, v1.0.1:** Patch 'Null' Z value handling.
- **February 2022, v1.0.2:** Patch to correct output Json schema.