

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 Xml2GeoJSON.py Python script, renamed from RSS2Json.py because of its extensive XML processing updates, is a Conversion routine designed to be run by the OverwriteFS.py script. The purpose of this script is to read XML data files like RSS or GeoRSS and convert then to GeoJSON feature collection files. At version 2.0, this script can process most any XML file. Geographic or GeoRSS data is recorded as GeoJSON Geometries. RSS or other XML content without a Geometry will be assigned a Point location of Latitude and Longitude of 0, 0 (if output as table not specified). If Longitude and Latitude fields are provided, using the 'xField' and 'yField' properties, this script will convert the values to Point features. Supports Depth or Elevation using 'zField' and Measures using 'mField'. Multiple GeoRSS feature types are supported (point, line, polygon, and box), creating one layer per geometry type. Box geometries are saved to Polygon layer. 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.

Version 2.0 includes the ability to “flatten” XML in Sub-Elements into fields.

Since RSS content includes a Publication date, this script will allow for detection and comparison of the Last Publication date/time to the current publication. 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 property. 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 RSS, GeoRSS, or other XML formatted file.

Capabilities and Workflow options

- XML Element names are used as field names. The Element text is used as the field data. Element Attributes can also be used to hydrate or augment a field. 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 XML will default to 'item' for RSS and GeoRSS, and 'feed' for ATOM and CAP. A user can specify a "Root Element" to access if the item list is named differently.
- Geometry construction can originate X, Y, Z and M ordinate 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
dateAsSeconds = False
publicationElement = <'publication' Element Name>
rootElement = <'item' Element Name>
flattenData = False
flattenNames = False
exclude = <Element Name>
exclude = <...>
trimOuterSpaces = True
ignorePrefix = False
allowNulls = True
sampleSize = 150
rowOffset = 0
rowLength = 0
outputExt = 'geojson' or existing
outputAsTable = False
xField = <X or Longitude Field Name>
yField = <Y or Latitude Field Name>
zField = <Z or Elevation Field Name>
mField = <M or Measure Field Name>
mIncrement = <Number value to add to Measure>
mOutput = True
zFactor = <Float or Integer Z Multiplier>
zOffset = <Float or Integer Z Incrementor>
zAbsolute = False
zOutput = True

[<input filename>.xml]
<xml Element Name> = <output field name> [<output field type> [<optional properties>]]
<xml 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 RSS file. This is compared to the next download to determine if there is a change in the publication data.
 - o '**dateAsSeconds**' is set as a True or False flag that tells the script what format the publication date is using when numeric is used. True, expect timestamp in integer Seconds (10 digits) and possibly Milliseconds as decimals. False, expect timestamp in integer Milliseconds (13 digits). Default is False.

- **'publicationElement'** stores the element name that should contain the publication date/time details. Default is best guess from list <'lastBuildDate', 'pubDate', 'updated', 'published'> or nothing if none of these are present.
- **'rootElement'** stores the XML list element name that should be processed to generate the GeoJSON data output. One item in the list per feature output. Default is 'item' for RSS and 'entry' for ATOM and CAP formats.
- **'flattenData'** is set as a True or False flag that tells the script to flatten all Sub-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. If the **'ignorePrefix'** property is False, the elements will include '<prefix>_<element name>' when the prefix is presented. Default is False
- **'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 False
- **'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
- **'ignorePrefix'** is set as a True or False flag that tells the script to ignore the Element 'prefix' value when creating the field names. XML (<prefix>:<element name>) is used to resolve Element naming conflicts between XML definitions. When set to False, the field names will include an underscore to separate the prefix and element name. 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. Element 'Cyclone:name' becomes 'Cyclone_name' as a field name. Ex. Multiple 'data_value' fields become 'data_value', 'data_value2', 'data_value3', and so on. Default is False
- **'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
- **'sampleSize'** specify number of rows to parse to determine field data types before start output. Default is 150
- **'rowOffset'** specify row number to start processing data. Default is 0
- **'rowLength'** specify number of rows to process. Default is 0

- **'outputExt'** specify output file extension to use. Default is 'geojson'
- **'outputAsTable'** is set as a True or False flag that tells the script to ignore the geometry details and output a Table instead of a Featureclass. Default is False
- **'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 * MUST be an Integer or Float <output field type>!**
- **'mField'** is set as a the [<output field name>](#) that will contain the Measure or 'M' value used to populate the Geometry field when no other Geometry value is available. If the row contains a Geometry, but does not include a M value, setting this property will add a M value to the existing Geometry! *** Note * MUST be an Integer or Float <output field type>!**
- **'mIncrement'** number value to add to Measurement ordinate before output. Default is 0 *** Note * MUST be an Integer or Float <output field type>!**
- **'mOutput'** is set as a True or False flag that controls the output of the Measure ordinate. Default is True
- **'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!**

- **'zAbsolute'** is set as a True or False flag that controls taking the Absolute value of the 'Z' coordinate value before adjusting. Default is False *** Note * MUST be an Integer or Float <output field type>!**
- **'zOutput'** is set as a True or False flag that controls the output of the 'Z' coordinate value. Default is True
- The **'[<input filename>.xml]'** or 'fields section' contains field layout details for the input file.
 - **'<xml element name>'** is Required and will contain the Element name identified in the XML 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 XML 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 XML. If the **'ignorePrefix'** flag is False, these element names can be included as **'<prefix>_<element name>'**. For computed fields, this can be 'NonOp' (non-operational) or some other nondescript name, as it has no real Element source. Can also reference 'Shape' ordinates as a source property by specifying 'SHAPE@X', 'SHAPE@Y', 'SHAPE@Z', 'SHAPE@M'. Same is true for record row id when 'ROWID@' is specified.
 - **'<output field name>'** is Required and will become the field name in the output file. *** 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 **<xml element name>**. A message is displayed if length exceeded!
 - **'<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"
 - **'[<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 [<xml 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!'
 - **'Attrib'** indicates the name of the [<xml element name>](#) 'attribute' to use as the field value. The attribute name can include a prefix and should be specified as '[<prefix>:<attribute name>](#)' if present or simply '[<attribute name>](#)' when not.
 - **'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!
 - **'AsSeconds'** is a Flag setting that does **NOT** use a 'value', it is a standalone field property! When included, expect numeric field value to be integer Seconds (10 digits) and possibly Millisecond decimals, otherwise expect Milliseconds (13 digits).
 - **'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.
- **'Abs'** will take the absolute value of this field. Use this when numeric 'sign' is not required or desired.
- **'Pow'** specifies a numeric value or Field data (highlighted) to raise this field to a power. Use this to multiply by itself 'x' times. Ex. If field value is '3': $3^1=3$, $3^2=9$, $3^3=27$, squared or cubed for instance.
- **'Root'** specifies a number value or Field data (highlighted) to take the root of this field. Use to return the inverse of 'Pow'. Ex. If field value is '27', '9', and '3': $27^{(1/3)}=3$, $9^{(1/2)}=3$, $3^{(1/1)}=3$, cubed or squared root for instance.
- **'Rand'** multiply this field by a random number between 0 and 1. Use this to generate a more random value as needed.
- **'Lambda'** specifies an in-line [Lambda](#) expression to use to process the field. Ex. Split field 'value' into an array and take the integer of second from last word: `lambda int(value.split()[-2])`

As an example, consider the following GeoRSS XML item data taken from Australia's Rural Fire Service Current Incidents site. This is an excerpt from a list of many items in the RSS feed.

Initially, the **Highlighted** Elements will be used to create fields in the GeoJSON, each being added to the output INI file automatically if there is no field section or fields defined. The Geometry 'point' and 'polygon' Elements in the source data will be used to create the Point and Polygon GeoJSON Layers. Note that since there are multiple Polygon entries for this item, the Polygon Feature will be built as a Multi-Part Polygon Feature. There will also be a Point Feature generated.

```

<item>
  <title>Darling Hills 2021 Ageclass Establishment Burn</title>
  <link>http://www.rfs.nsw.gov.au/fire-information/fires-near-me</link>
  <description><![CDATA[
ALERT LEVEL: Advice <br />LOCATION: Wonga Road, Blenheim State Forest <br />COUNCIL AREA: Oberon <br />STATUS:
Under control <br />TYPE: Hazard Reduction <br />FIRE: Yes <br />SIZE: 117 ha <br />RESPONSIBLE AGENCY: Forestry
Corporation of NSW <br />UPDATED: 2 Sep 2021 10:19]]></description>
  <category>Advice</category>
  <pubDate>Thu, 02 Sep 2021 06:36:54 GMT</pubDate>
  <guid>https://incidents.rfs.nsw.gov.au/api/v1/incidents/402852</guid>
  <point xmlns="http://www.georss.org/georss">-33.6316293959999 149.871711731</point>
  <polygon xmlns="http://www.georss.org/georss">-33.6249889009999 149.86787395 -33.62498692 149.867867184 -
33.624942914 149.867989772 -33.6249889009999 149.86787395</polygon>
  <polygon xmlns="http://www.georss.org/georss">-33.62498692 149.867867184 -33.62523333 149.867180756 -
33.625216552 149.867180756 -33.6248834269999 149.86751388 -33.62498692 149.867867184</polygon>
  <polygon xmlns="http://www.georss.org/georss">-33.62523333 149.867180756 -33.62526414 149.867180756 -
33.6249889009999 149.86787395 -33.6255734709999 149.869869547 -33.625359319 149.871630348 -33.626620434
149.874033603 -33.626905968 149.87515195 -33.628119494 149.875009182 -33.629547171 149.87607994 -33.630261009
149.876984135 -33.632521497 149.876888956 -33.6330687729999 149.877531411 -33.6298327059999 149.877531411 -
33.629761322 149.878768731 -33.630570339 149.880267791 -33.632711854 149.880672299 -33.634781986 149.881909619 -
33.6359003319999 149.880243997 -33.638065641 149.879316007 -33.637399393 149.87726967 -33.637232829 149.87665101 -
33.6379942569999 149.876175118 -33.640552178 149.874652263 -33.6391720899999 149.874295344 -33.6370543699999
149.872867668 -33.635888434 149.871749321 -33.635364953 149.869845752 -33.633770714 149.869607806 -33.632628573
149.868941558 -33.630082549 149.866109998 -33.625276038 149.867061783 -33.62523333 149.867180756</polygon>
</item>

```

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

```

[properties]
lastPublicationDate = 2021/09/04 07:40:23
rootElement = item

[NSW_Rural_Fire_Service.json]
title = title
link = link
description = description
category = category
pubDate = pubDate
guid = guid

```

Customization 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 'description' field data in this example very closely and you'll find that there is a wealth of information hidden away if left as a single text description field. We can redefine the INI file to pull out as much of this detail as possible!

Isolating just the description field in the input data below, we can see the available content highlighted in yellow. The 'CDATA' or Character Data definition in the XML is handled internally by the

XML processor, so you will not have to adjust any of the extraction properties to work with this data, simply ignore it.

```
<description><![CDATA[
ALERT LEVEL: Advice <br />LOCATION: Wonga Road, Blenheim State Forest <br />COUNCIL AREA: Oberon <br />STATUS:
Under control <br />TYPE: Hazard Reduction <br />FIRE: Yes <br />SIZE: 117 ha <br />RESPONSIBLE AGENCY: Forestry
Corporation of NSW <br />UPDATED: 2 Sep 2021 10:19]]></description>
```

Starting with the initial INI file, we can alter it by adding new fields that are derived from the highlighted data in the description Element content. Take notice of the Size field conversion from Hectares to Acres, this takes advantage of the scripts ability to compute values using simple Math.

```
[properties]
lastPublicationDate = 2021/09/04 07:40:23
rootElement = item
trimOuterSpaces = True

[NSW_Rural_Fire_Service.json]
title = title
link = link
description = description
description = alertLevel text Start ALERT%20LEVEL: End <br
description = location text Start LOCATION: End <br
description = councilArea text Start COUNCIL%20AREA: End <br
description = status text Start STATUS: End <br
description = type text Start TYPE: End <br
description = fire text Start FIRE: End <br
description = size float Start SIZE: End %20 Mult 2.471
description = responsibleAgency text Start RESPONSIBLE%20AGENCY: End <br
description = updated date Offset -20
category = category
pubDate = pubDate
guid = guid
```

The resulting GeoJSON data for this example, showing just the Point feature, looks like this:

```

{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "category": "Advice",
        "description": "ALERT LEVEL: Advice <br />LOCATION: Wonga Road, Blenheim State Forest <br />COUNCIL AREA: Oberon <br />STATUS: Under control <br />TYPE: Hazard Reduction <br />FIRE: Yes <br />SIZE: 117 ha <br />RESPONSIBLE AGENCY: Forestry Corporation of NSW <br />UPDATED: 2 Sep 2021 10:19",
        "alertLevel": "Advice",
        "location": "Wonga Road, Blenheim State Forest",
        "councilArea": "Oberon",
        "status": "Under control",
        "type": "Hazard Reduction",
        "fire": "Yes",
        "size": 289.107,
        "responsibleAgency": "Forestry Corporation of NSW",
        "updated": "2021-09-02 10:19:00",
        "guid": "https://incidents.rfs.nsw.gov.au/api/v1/incidents/402852",
        "link": "http://www.rfs.nsw.gov.au/fire-information/fires-near-me",
        "pubDate": "Thu, 02 Sep 2021 06:36:54 GMT",
        "title": "Darling Hills 2021 Ageclass Establishment Burn"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [149.871711731, -33.6316293959999]
      }
    }
  ]
}

```

Command-Line Usage / Execution

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

```
'Python Converters\Xml2GeoJSON.py <sourceFilename> [<checkPublication> [<verbose>]]'
```

Command-Line Input Parameters

- Available Input Parameters

<sourceFilename>: (required) String Path and or Filename of source XML 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.

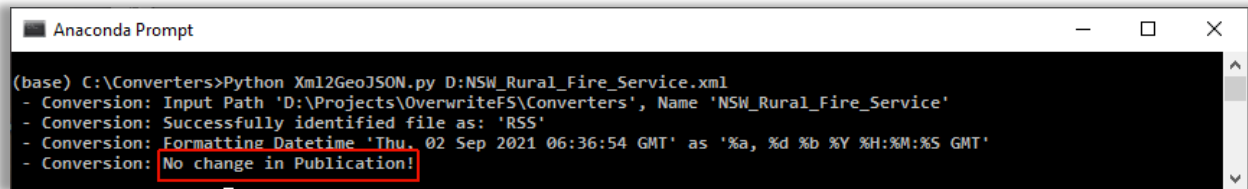
Default is True

<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

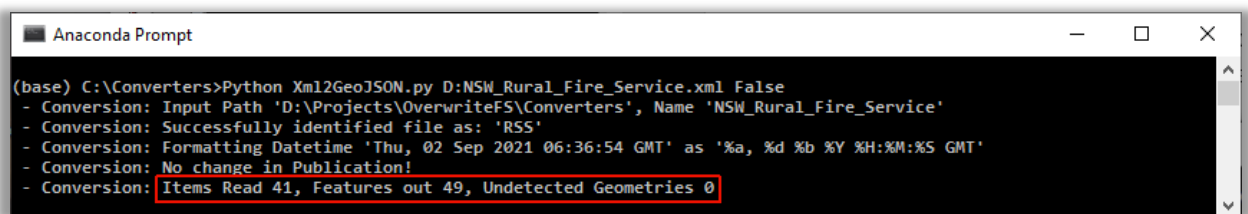
- Execution with just the source Filename. Script shows input file path, name, and detected type. It also shows the evaluated Publication Date/Time format and value, reporting there has been no change to the data since last run.



```

(base) C:\Converters>Python Xml2GeoJSON.py D:\NSW_Rural_Fire_Service.xml
- Conversion: Input Path 'D:\Projects\OverwriteFS\Converters', Name 'NSW_Rural_Fire_Service'
- Conversion: Successfully identified file as: 'RSS'
- Conversion: Formatting Datetime 'Thu, 02 Sep 2021 06:36:54 GMT' as '%a, %d %b %Y %H:%M:%S GMT'
- Conversion: No change in Publication!
  
```

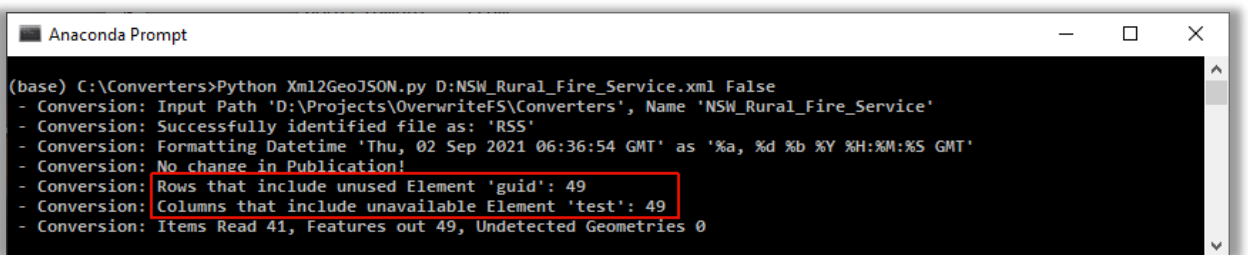
- Example run that overrides the 'checkPublication' value, set to False and ignoring the PubDate. This forces the routine to process the data even if there has been no change. The additional detail reports the number of rows read in and written out. Since the GeoRSS entries can contain multiple Point/PolyLine/Polygon Geometries on the same record, multiple features will often be created.



```

(base) C:\Converters>Python Xml2GeoJSON.py D:\NSW_Rural_Fire_Service.xml False
- Conversion: Input Path 'D:\Projects\OverwriteFS\Converters', Name 'NSW_Rural_Fire_Service'
- Conversion: Successfully identified file as: 'RSS'
- Conversion: Formatting Datetime 'Thu, 02 Sep 2021 06:36:54 GMT' as '%a, %d %b %Y %H:%M:%S GMT'
- Conversion: No change in Publication!
- Conversion: Items Read 41, Features out 49, 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 report any field that has trouble being processed.



```

(base) C:\Converters>Python Xml2GeoJSON.py D:\NSW_Rural_Fire_Service.xml False
- Conversion: Input Path 'D:\Projects\OverwriteFS\Converters', Name 'NSW_Rural_Fire_Service'
- Conversion: Successfully identified file as: 'RSS'
- Conversion: Formatting Datetime 'Thu, 02 Sep 2021 06:36:54 GMT' as '%a, %d %b %Y %H:%M:%S GMT'
- Conversion: No change in Publication!
- Conversion: Rows that include unused Element 'guid': 49
- Conversion: Columns that include unavailable Element 'test': 49
- Conversion: Items Read 41, Features out 49, 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

- **September 2021, v1.0.0:** Initial public release.
- **November 2021, v2.0.0:** Renamed to Xml2GeoJSON.py because of the extensive updates made to reading generic XML files, and to better clarify the GeoJSON output. Added INI file properties for '[rootElement](#)', '[flattenData](#)', '[flattenNames](#)', '[exclude](#)', '[trimOuterSpaces](#)', '[ignorePrefix](#)', '[allowNulls](#)', '[xField](#)', '[yField](#)', '[zField](#)', '[zFactor](#)', and '[zOffset](#)'. Added Optional field property '[Attrib](#)', '[Case](#)', '[AllowNulls](#)', and '[DoNotSave](#)'. Updated Extraction logic to handle properties as a linear chain of operations and added operators '[Concat](#)', '[Add](#)', '[Sub](#)', '[Mult](#)', and '[Div](#)' with the ability to leverage processed fields as input. Default field property also allows for leveraging processed field values as input. Detect max field name length. Can create Point Geometries from field values or augment Point and Multi-Point Z elevations using field value, offset, or multiplier.
- **December 2021, v2.0.1:** Patch 'Null' Z value handling.
- **February 2021, v2.0.2:** Patch Json schema output.
- **August 2024, v1.1.0:** Fixed Root Element identification. Json output format issue. Added initial Field type identification. Enhanced date field epoch handling by including '[AsSeconds](#)' flag. Patch '[Length](#)' extract function. Added hard stop on INI config issues. Added '[publicationElement](#)', '[dateAsSeconds](#)', '[sampleSize](#)', '[rowOffset](#)', '[rowLength](#)', '[outputExt](#)', '[zAbsolute](#)', '[zOutput](#)', '[mField](#)', '[mIncrement](#)' and '[mOutput](#)' to Properties section. Extended Z manipulations to polyline and polygon features. Altered x/y/z/[mField](#) Properties to allow assignment from a field, not just set as a default. Altered [zOffset](#) and [zFactor](#) to allow field specification. Added '[abs](#)', '[pow](#)', '[root](#)', '[rand](#)' and '[lambda](#)' to field extraction options. Added access to Point ordinate values with 'SHAPE@X/Y/Z/M' and 'ROWID@' element names for fields. Added '[outputAsTable](#)' Property. Updated to generate initial INI file if not found, reporting candidates for [rootElement](#) picking best.