

Table of Contents

Overview 2

Requirements 2

Capabilities and Workflow options..... 3

Input file INI structure..... 4

Command-Line Usage / Execution..... 9

Command-Line Input Parameters 9

Command-Line Examples..... 10

Guidance, Limitations, and Known Issues..... 11

Release History 12

Overview

The Rss2Json.py Python script is a Conversion routine designed to be run by the OverwriteFS.py script. The purpose of this script is to read RSS or GeoRSS formatted XML files and convert then to GeoJson. Geographic or GeoRSS data is recorded as GeoJson Geometries. RSS content without a Geometry will be assigned a Point location of Latitude and Longitude of 0, 0. 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 a Service. The INI file also allows for data extraction from the field value before saving. Example would be to extract properties from Comment or Description text in the data and save as separate 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 file.

Capabilities and Workflow options

- XML Element names are used as field names. The Element text is used as the field data. If no text is available and an Attribute is available, the first Attribute data element will be used. 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 a field Type default will be assigned.
- Fields can be created from a source Element data by extracting content from the data using the INI field definition.
- Reading a list of Items from the XML is limited to 'item' for RSS default or 'feed' for ATOM and CAP default. Others could be included in the future, along with a user provided item list Element name.

Input file INI structure

When reading the input 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 format structured as follows:

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

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

- The **'properties'** section contains feed specific processing details that are carried from run to run.
 - o The '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.
- The **'<input filename>.json'** section contains field layout details for the input file.
 - o **'<xml element name>'** is Required and will contain the Element name identified in the XML data. The Value for the field will be pulled from this Element's content. Elements that do not exist in the RSS item data can be defined and added to the output. These can be schema additions for layer computation or stand ins for fields that have not yet been created in the RSS.
 - o **'<output field name>'** is Required and will become the field name in the output file. Default is the Element name.
 - 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.** Though, if the type is 'date', the Rss2Json script will attempt to interpret the field contents as Date/Time. Types include:
 - **'text'** is a string character array field. This is the Default field type!
 - **'integer'** is a 32-bit integer number field
 - **'float'** is a Double-Precision floating point field
 - **'date'** is a Date/Time field

- '[<optional properties>]' are of course optional, but if used you must specify a Field Type property first! Entries here are processed as space delimited pairs of space separated '<setting> <value>' entries supporting the following settings. *** 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 fields. 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'
 - **'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. If the default is not provided, a field type specific value will be used, '0' for integer; '0.0' for float; an empty string is used for text; and '1970/01/01T00:00:00' for date. Ex. 'default This%20is%20a%20test!'

Element data Extraction can be handled by using the following field property entries. A starting and ending position or starting and ending search text can be provided to define the data you wish to 'extract' from the Element's content.

- **'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. Can be coupled with 'offset' to begin searching at an offset position. Can be coupled with length to search for the starting text and then extract a given length of characters. 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. Can be used with offset to begin searching for end starting at the offset provided. Used without any other extraction properties will result in truncating content to the RIGHT of this end! Default is length of Element content remaining.

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. The Geometry 'point' and 'polygon' Elements 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

[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 fields can be created by extracting content from existing fields. As an example, examine the 'description' field data closely and you'll find that there is a wealth of information that is hidden away if left sitting in the description field alone.

We can redefine the INI file to pull out as much of the description details 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.

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

[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 Default 0.0 Start SIZE: End ha
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": "117",
        "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\Rss2Json.py <sourceFilename> checkPublication=True verbose=True'
```

Command-Line Input Parameters

- Available Input Parameters

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

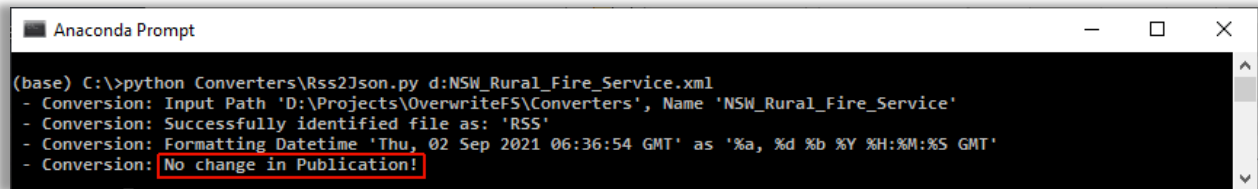
Default is True

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

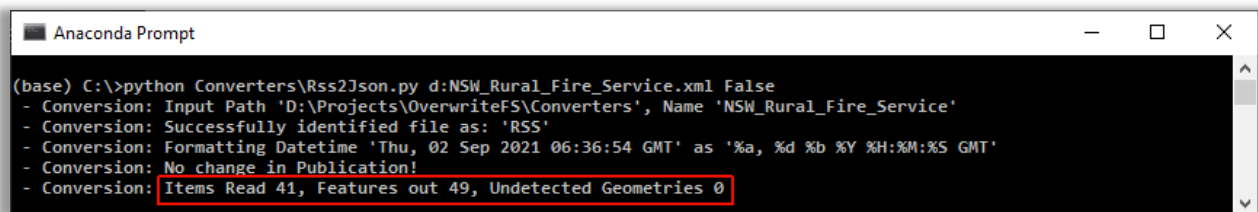
Command-Line Examples

- 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.



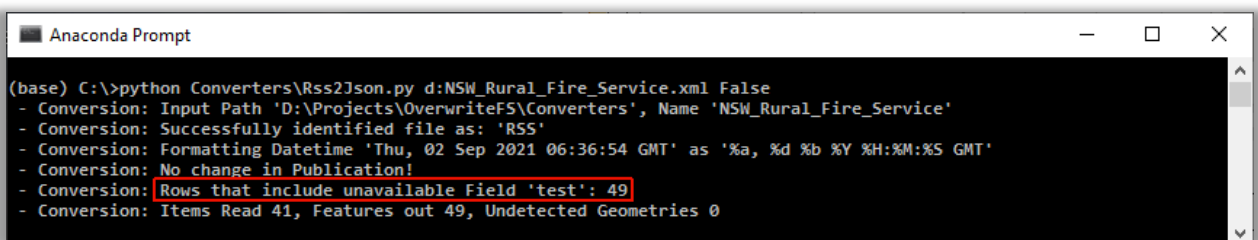
```
Anaconda Prompt
(base) C:\>python Converters\Rss2Json.py d:NSW_Rural_Fire_Service.xml
- Conversion: Input Path 'D:\Projects\OverwriteFS\Converters', Name 'NSW_Rural_Fire_Service'
- Conversion: Successfully identified file as: 'RSS'
- Conversion: Formatting Datetime 'Thu, 02 Sep 2021 06:36:54 GMT' as '%a, %d %b %Y %H:%M:%S GMT'
- Conversion: No change in Publication!
```

- Execution that overrides the 'checkPublication' value, setting it 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. The dual Point/Polygon Geometries on some features is the difference in the count.



```
Anaconda Prompt
(base) C:\>python Converters\Rss2Json.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 also display a row count for any field defined that does not have an Element, no source data, typical for any ad-hoc field added by the user or where the Element is missing from the data source. Here, you can see a 'test' field that was added, pointing to a test Element that does not exist. This script will report any field that has trouble being processed.



```
Anaconda Prompt
(base) C:\>python Converters\Rss2Json.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 unavailable Field '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!
- **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

- **Sep 2021, v1.0.0:** Initial public release.