Table of Contents

Overview

The OverwriteFS Python script is designed to run as a standalone routine OR it can be ingested by Python scripts or Python Notebooks for use in your own custom tools and workflows.

Once a Hosted Feature Service has been created in ArcGIS Online or on your Enterprise Portal, you can leverage this script to regularly overwrite (re-publish) the service using your updated Service File (originally used to Publish the service). If your source data resides on a public internet location, you can use this script to download the updated file data and overwrite your service. Restrictions may apply if a Tile, Vector Tile, OGC, or WFS service has been added to the Feature Service or View.

Launch as a stand-alone process from the Command Prompt or create a Task in the Windows Task Scheduler to execute at specific times.

Import into your own Python script or Python Notebook as a Module and integrate the overwrite functions in your workflows.

Requirements

1. Python 3.x or Python Notebook
2. ArcGIS Python API 1.5.1+
3. Pre-existing ArcGIS Python API ‘GIS’ module connection ‘Profile’, see Python API Documentation for options and examples.
   a. If an ArcGIS Pro connection is needed, ArcGIS Pro and Arcpy will also be required.
4. Pre-existing File-Based Hosted Feature Service with related File Item used during publishing.
5. Write access to user’s Temp folder, to store intermediate item configuration files, as a backup, during Overwrite process.

Capabilities and Workflow options

- Touch a Service and/or View Item's "Last Updated" property, visible on the Item's detail page:
    - Updating the Service Item without modifying any properties can be used as proof that your update process has run, even if it hasn't changed any data.

- Overwrite a Service using updated local data or Internet resource:
    - Overwriting or re-Publishing is one way to replace the entire data set contained in a Service, renewing its contents.
    - Requires the Service to have a related File Item (created during initial Publishing).
    - Will likely disrupt user access during the overwrite process. The larger the data set is, the longer the Service will be inaccessible.
    - Overwrite action will also update the Data visible to any View created from the Service.

- Minimize user access disruption during overwrite by using "Layer Swap" capability:
    - Leverage '***-SwapLayers***' Action Switch to initiate with/without Overwrite.
    - Leverage '***-GetTarget***' Action Switch to report Target Service for Layer Swap.
    - Manages 2 identical Services, 'A' and 'B', using one Service View.
    - Update Service 'B' while users access Service 'A', referenced by View.
    - Swap Layers in View from Service 'A' by pointing to Service 'B'.
    - Allowing overwrite of Service 'A'.
    - Repeat…

- Enhance functionality by invoking a Conversion tool Post-Download and Pre-Service-Update:
    - Ex: Includes an 'Rss2Json' conversion Module that takes an RSS or GeoRSS file and converts it to GeoJson for upload to ArcGIS Online.
    - Leverage this '***-Convert***' Option Switch to customize or tailor the output as needed.

- Manage Service-to-View Relationships, required by "Layer Swap" capability:
    - Leverage '***-AddRelated***', '***-RemoveRelated***', or '***-ListRelated***' Action Switches to add, remove, or display Service relationships to a View.

- Restore lost File Item Relationship to existing service:
    - A File Item is required when Overwriting a service. A File Item is the associated Item that was initially uploaded/created during the Service Publication process. This Item contains the schema and data used to initially create and hydrate the Service. If this Item is removed post-publishing, you will lose the ability to Overwrite the Service Item.
    - If a local copy of the uploaded file still exists, you can recreate the missing or deleted File Item by uploading the file as a new Item, and then reestablish the relationship it once had to the Service, allowing you to overwrite the Service once again.

Command-Line Usage / Execution

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

'Python <path>\OverwriteFS.py <profile> <Service Item Id> <Service Title>'

Or for usage help, type:

'Python <path>\OverwriteFS.py -h'

Command-Line Response

When returning from execution, the script will set the Exit Code (or Errorlevel on Windows) to report the outcome of the run.

- Exit code 1 = Failure
- Exit code 0 = Success, update was made
- Exit code -1 = No changes were made

Command-Line Validation and Input Error Messages

- Alphabetical list of common Input and Validation Error responses, all resulting in an Error exit outcome of '1' returned to the command-line.

Data Conversion Failed, Error: '<e>': The '*-Covert*' option failed to convert the downloaded data because of an error specified by <e>.

Feature Service Item is a 'View', cannot Overwrite: The specified Service Item is a Hosted Feature View, which cannot be Overwritten. Check that the correct Item is being used, verify using ArcGIS Online if needed.

Feature Service Item is NOT a 'View': The specified View Item is NOT a Hosted Feature View. The script is unable select a Target Service to Overwrite. Check that the correct Item Id is being used, verify using ArcGIS Online if needed.

Feature Service Item Title does NOT match the specified Title: '<xyz>', Found: '<abc>': The specified Title does not match the Title of the specified Item Id. Check for misspellings, you may also need to surround the title with Double-Quotes if it includes spaces. The command prompt uses spaces to identify each input parameter, quoting them will treat the phrase as an input parameter.

Feature Service View requires 2 Related Feature Services or 2 Related Feature Views, found: {n}: The specified View Item has more or less than 2 Related Hosted Services or Views. Swap Layers action requires 2 and only 2 related Items. Check that the correct Item Id is being used, verify using ArcGIS Online if needed.

Insufficient Input Parameters: The minimum number of input parameters were not provided, resulting in the script returning this message along with Usage details.

Item is NOT a Feature Service 'View', cannot Swap Layers: The specified Service Item is NOT a Hosted Feature View. You can only Swap Layers of a View, no other Item type supports this action! Check that the correct Item Id is being used, verify using ArcGIS Online if needed.

Item Type is NOT a 'Feature Service': The specified Service Item is NOT a Hosted Feature Service or View. The script cannot Overwrite or Swap Layers on any other Item types. Check that the correct Item is being used, verify using ArcGIS Online if needed.

Loading Converter '<xyz>' Failed, Error: ' Failed to evaluate Parameter '<a>', Error: '<e>'': The specified “Converter” Module named <xyz> could not evaluate parameter <a> because of an error reported by <e>. If a String or File specifier is provided, you may need to include Quotes around the String. Review the Module documentation for details and requirements.

Loading Converter '<xyz>' Failed, Error: ' Failed to import module '<xyz>', Error: ' Function 'convert' has NO parameters, minimum of 1 is required'': The ‘convert’ function in the specified “Converter” Module named <xyz> does not include parameters. At least one is required to pass along the download path and filename for processing. Review the Module name specified and check that it conforms to the requirements.

Loading Converter '<xyz>' Failed, Error: ' Failed to import module '<xyz>', Error: ' Module is missing 'convert' function'': The specified “Converter” Module named <xyz> does not contain a Function called ‘convert’. Review the Module name specified and check that it conforms to the requirements.

Loading Converter '<xyz>' Failed, Error: ' No Parameters required, <n> specified': The ‘convert’ function in the specified “Converter” Module named <xyz> only has one parameter (the download File specifier), no additional parameters are supported, yet <n> were included in the convert request. Review the Module name specified and check that it conforms to the requirements.

Loading Converter '<xyz>' Failed, Error: ' Too many Parameters specified, <n>, only need values for ('<arg>, <…>')': The ‘convert’ function in the specified “Converter” Module named <xyz> has multiple parameters, listed by <arg>, but more parameters were specified. A total of <n> were specified. Review the Module name specified and check that it conforms to the requirements.

Login failed, please verify Profile: The provided Profile could not be used to log into ArcGIS Online or Enterprise Portal. Check the Profile using the Python API to resolve the issue and try again.

Missing Associated Service Data Item or datafile Item 'name': The specified Service Item is missing its associated File Item, or, it is missing the ‘Service2Data’ relationship to the File Item. Check that the correct Item Id is being used, verify using ArcGIS Online if needed. If the File is located, check for missing relationship by using the ‘***-ListRelated***’ Action Switch for specified Service Item. Ideally, this item should ‘Rely on Data Item’ you found! If not, the ‘***-AddRelated***’ Action Switch can be used to restore the lost relationship.

Missing Update File or Invalid Switch: ‘<xyz>’: One or more of the additional parameters were incorrect or not recognized.

Mutually exclusive parameters are set, cannot Preserve and Ignore Service/View Properties: The '**-NoProps**' and '**-PersistProps**' parameters were both specified (enabled). The script cannot Set and Not Set the properties on the specified Service Item at the same time.

No Data Service Target available: No Hosted Feature Service Items could be Targeted for the Overwrite action on the specified View Item. Overwrite action cannot be performed. Check that the correct Item Id is being used, verify using ArcGIS Online if needed.

Overwrite on Service is NOT allowed, a dependent View or Service has Change Tracking Enabled: The specified Service Item cannot be Overwritten because a dependent Service has 'Change Tracking' enabled. Change Tracking is a dependent condition when a '**Tile**' Service (not a Vector Tile Service) is created from the Hosted Service or View. Overwriting the Service will disrupt or break the Service! Check that the correct Item Id is being used, verify using ArcGIS Online if needed.

Overwriting on Service is NOT allowed, a dependent OGC or WFS Service exists: The specified Service Item includes a dependent Open Geospatial Consortium (OGC) or Web Feature Service (WFS) Service Item created from it. You cannot Overwrite the Service if one of these dependencies exist, it will break the OGC or WFS Service!

Password missing for user '<abc>' in Profile '<xyz>': The stored input Profile does not contain a password. Check the Profile using the Python API to resolve the issue and try again.

Related View does NOT contain a Data Source: The specified View Item's Related View does NOT contain a Parent Hosted Feature Service. The script is unable select a Target Service to Overwrite. Check that the correct Item Id is being used, verify using ArcGIS Online if needed. If the Related Hosted Feature View does not in fact have a Parent Service, the '***-AddRelated***' Action Switch can be used to reconnect to the proper Service Item. * Use extreme Caution!

Swapping Layers is NOT allowed, a dependent OGC or WFS Service exists: The specified View Item includes a dependent Open Geospatial Consortium (OGC) or Web Feature Service (WFS) Service Item created from it. You cannot Swap Layers of the View if one of these dependencies exist, it will break the OGC or WFS Service!

Target Feature Service is missing a file Data Source: The Targeted Hosted Feature Service selected by the Swap Layers with Overwrite action is missing an associated File Item. The Overwrite action cannot be performed. Check that the correct Item Id is being used, verify using ArcGIS Online if needed.

Unable to Locate specified Item: '<xyz>': The specified Service Item cannot be found using the Item Id. Check that the correct Item Id is being used, verify using ArcGIS Online if needed.

Unable to locate Update File or Folder '<xyz>': The update File or Path provided cannot be found. The specified file may not exist or is incorrectly specified, or the script may not have access to the resource.

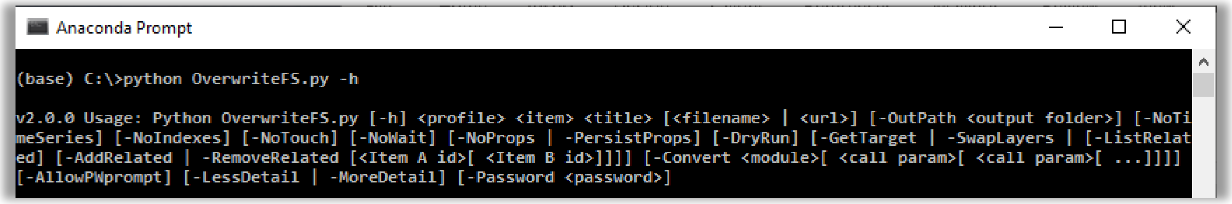
Update Filename '{abc}' does NOT Exist: The specified Service Item update Filename cannot be found. Check that the correct path and Filename is being used.

Update Filename '{abc}' does NOT match Original Filename used to Publish Service: '{xyz}': The specified Service File Item Filename, '{abc}', does not match the Filename, '{xyz}', used to Publish the Original Service Item.

Update Filename '{abc}' is NOT a File: The specified Service Item update Filename is not file. Check that the correct path and Filename is being used.

Update Filename '{abc}', Originally Published with Service, CANNOT be found in Folder: '{xyz}': The specified Service File Item's Filename, '{abc}', cannot be found in the specified folder, '{xyz}'. Verify that the file location has been specified correctly.

Command-Line Input Parameters



```
(base) C:\>python OverwriteFS.py -h

v2.0.0 Usage: Python OverwriteFS.py [-h] <profile> <item> <title> [<filename> | <url>] [-OutPath <output folder>] [-NoTi
meSeries] [-NoIndexes] [-NoTouch] [-NoWait] [-NoProps | -PersistProps] [-DryRun] [-GetTarget | -SwapLayers | [-ListRelat
ed] [-AddRelated | -RemoveRelated [<Item A id>[ <Item B id>]]]] [-Convert <module>[ <call param>[ <call param>[ ...]]]]
[-AllowPWprompt] [-LessDetail | -MoreDetail] [-Password <password>]
```

- Available Input Parameters

-h: (optional) Action Switch that triggers 'usage' display then exits.

<profile>: (**required**) A pre-generated Python API connection profile name (with stored connection details) which has administrative access to the Service Item you wish to update. You can also specify the string text 'pro', telling the script to use the active ArcGIS Pro connection setting for the user running the script.

<item>: (**required**) Service Item Id of the Service you wish to Update or interact with.

<title>: (**required**) Service Item 'title', used to verify that the item Id is the correct item you want to update. *** Please Note *** You will need to wrap the title text in Double-Quotes if spaces are included in item title, allowing DOS to accept a title *phrase* as a single input parameter. Ex. "MODIS C6 Global 24h"

<filename>: (optional) file path and/or name of file to upload, or URL to online data file. If 'path' only, script will search path for original filename used to update Service.

Default: Empty string, only touch Item and/or Views to refresh last update date.

<url>: See ***<filename>***

-OutPath: (optional) Option Switch setting output file Path used to store ***<url>*** file download and Service property backup files during run.

Default: User's Temporary folder.

-NoTimeSeries: (optional) Option Switch instructing function not to touch Time Extent of Time Series enabled Layers in Service and related Views.

Default: Touch Layer Time details on Item and related Views, reflecting new data.

-NoIndexes: (optional) Option Switch instructing function not to recreate missing Layer field Indexes on Service.

Default: Recreate indexes if they are missing after an Overwrite action.

-NoTouch: (optional) Option Switch instructing function not to update the 'lastUpdated' item property for the Service or related Views unless an update has been made.

Default: Update, or "touch", the 'lastUpdate' property on the Service item and Views, even when an update has not been made. Signifies that the process has been run.

-NoWait: (optional) Option Switch instructing function not to wait for re-application of properties like Layer Optimization to complete before continuing the Overwrite or Layer Swap action. When enabled, function will report condition and supply a URL that can be used for manual status review.

Default: Function will wait for properties like Layer Optimization to be re-applied before proceeding to next processing 'step' in Overwrite or Layer Swap workflow.

-NoProps: (optional) Option Switch instructing function NOT to Re-Apply Service or View properties following a successful update. This Defaults Service or View back to its Published state, mimicking the operation of Version 1.4.4 and reducing workload!

* Warning * Any Post-Publishing changes WILL BE LOST!

* Caution * If set to True, '***-PersistProps***' setting CANNOT be set to True!

Default: Re-Apply properties and do not persist Backup file beyond successful update.

-PersistProps: (optional) Option Switch instructing function to retain Service or View property Backup File after a successful Overwrite and property restoration. Will be used in subsequent Overwrite actions while option persists.

* Caution * If set to True, '***-NoProps***' setting CANNOT be set to True!

Default: Do not persist Backup file beyond successful update.

-DryRun: (optional) Option Switch instructing function to step through process WITHOUT updating the Service.

Default: Update or Touch the Service and Item.

-SwapLayers: (optional) Action Switch instructing function to Swap Layers in View, point all Layers to Target or newly Overwritten Feature Service. Used by A/B Feature Service enabled View, whereby the View is Related to Two Feature Services, allowing the View's Layers to be swapped out to point to the matching Layers of the newly updated Feature Service.

Default: Overwrite action when '***<filename>***' or '***<url>***' specified and no other Action Switches included.

-GetTarget: (optional) Action Switch instructing function to report the selected Feature Service Item Id, Title, and File Item upload Filename that should be used for the next update target. Leveraged by A/B View enabled Services to select inactive Feature Service that should be overwritten next.

Default: Overwrite action when '***<filename>***' specified and no other action switches included.

-ListRelated: (optional) Action Switch instructing function to List all Items related to '***<item>***' View or Service.

Default: Overwrite action when '***<filename>***' specified and no other action switches included.

-AddRelated: (optional) Action Switch instructing function to Add specified Service Item(s) to View '***<item>***', limit two Item Ids as Related Services. Related Services are required for '***-SwapLayers***' action to switch Layers of View so they point to Layers referenced by Related 'target' Feature Service.

Default: If No Item IDs specified, action will be to '***-ListRelated***' relationships for '***<item>***'.

-RemoveRelated: (optional) Action Switch instructing function to Remove specified A/B Service Items from View '***<item>***'. Related Feature Services are required for '***-SwapLayers***' to target and switch Layers of '***<item>***' View so they point to same Layers referenced by Related 'target' Feature Service.

Default: If No Items specified, action will be to '***-ListRelated***' relationships for '***<item>***'.

<Item A id> and

<Item B id>: (optional, but at least one is required when using '***-AddRelated***' or '***-RemoveRelated***' Action Switch). The unique Item Identifier for the Content Item managing the Service. Ex: Id for 'Active Hurricanes, Cyclones, and Typhoons' is: 248e7b5827a34b248647afb012c58787

-Convert: (optional) Action Switch instructing function to import specified Python Module and use it to transform the source data before Overwriting the Hosted Feature Service. Download file is passed to Function as first parameter. When available, a True or False Boolean value, based on 'detail', is passed to 'verbose' parameter.

*** Note *** Only available during a Overwrite action.

Default: No conversion of the input data will take place.

<module>: (optional but required when using '***-Convert***' Action Switch) A Case-Sensitive Python Module (or script) name to leverage for data conversion prior to updating Service. Supports Python 'Dot' notation for Folder/Module/Class path access to 'convert' Function.

*** Note *** Path is **RELATIVE** to 'Converters' folder in OverwriteFS.py script location!

Ex: [<folder>.]<module>[.<class>] where path and class are optional

<call param>: (optional) Additional call parameters to pass to '***-Convert***' ***<module>*** function. Only need to include parameters that follow the download file string specifier in 'convert' Function. Does NOT support Keyword Arguments, positional ONLY!

-AllowPWprompt: (optional) Option Switch instructing function to Allow use of an undefined user account password in the Profile, allowing Python API to prompt for user entry.

*** Note *** *Not recommended for use during Unattended execution!*

Default: If Profile Password is not defined, function will report the condition and Exit.

-LessDetail: (optional) Option Switch instructing function to only Display major steps and error responses.

Default: Display step by step processing detail, or '***-MoreDetail***' if set.

-MoreDetail: (optional) Option Switch instructing function to Display maximum Diagnostic detail and error responses.

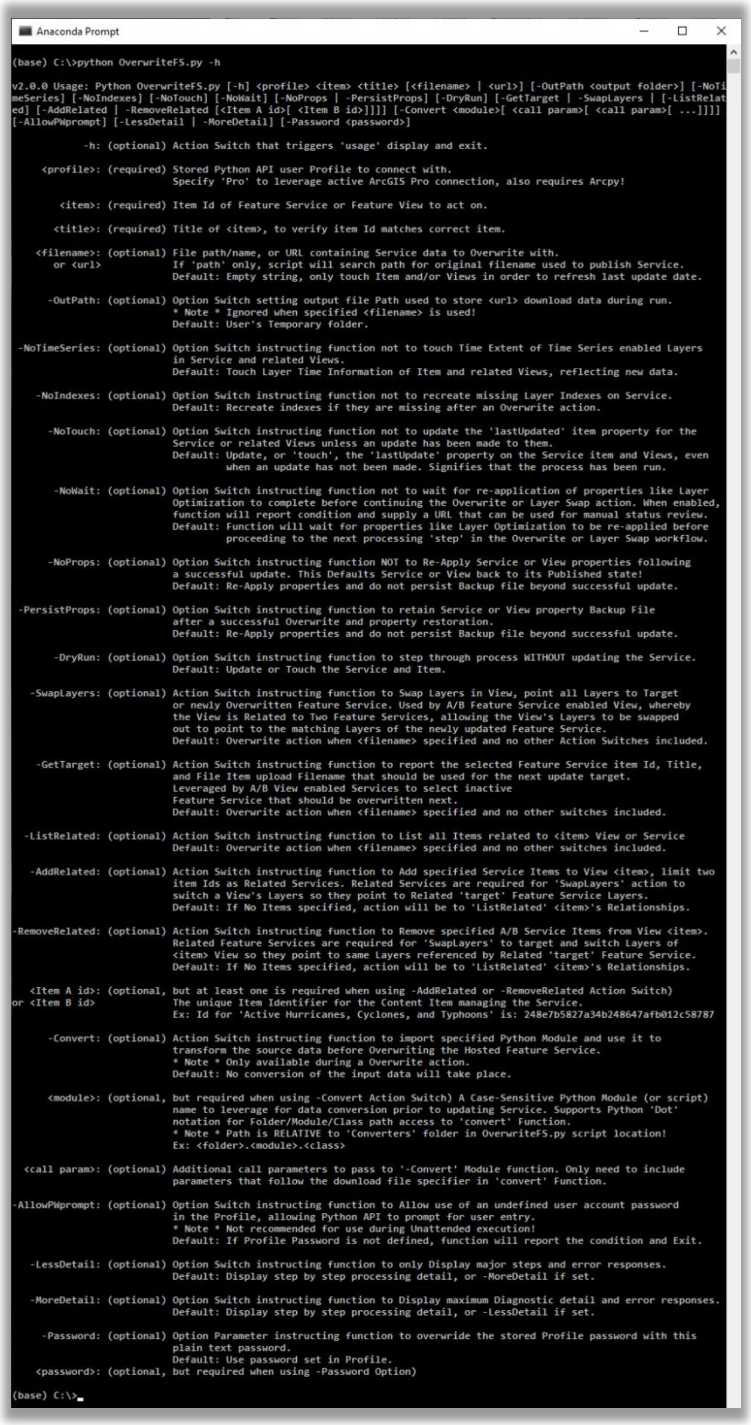
Default: Display step by step processing detail, or '***-LessDetail***' if set.

-Password: (optional) Option Parameter instructing function to overwrite the stored Profile password with this plain text password.

Default: Use password set in Profile.

Command-Line Examples

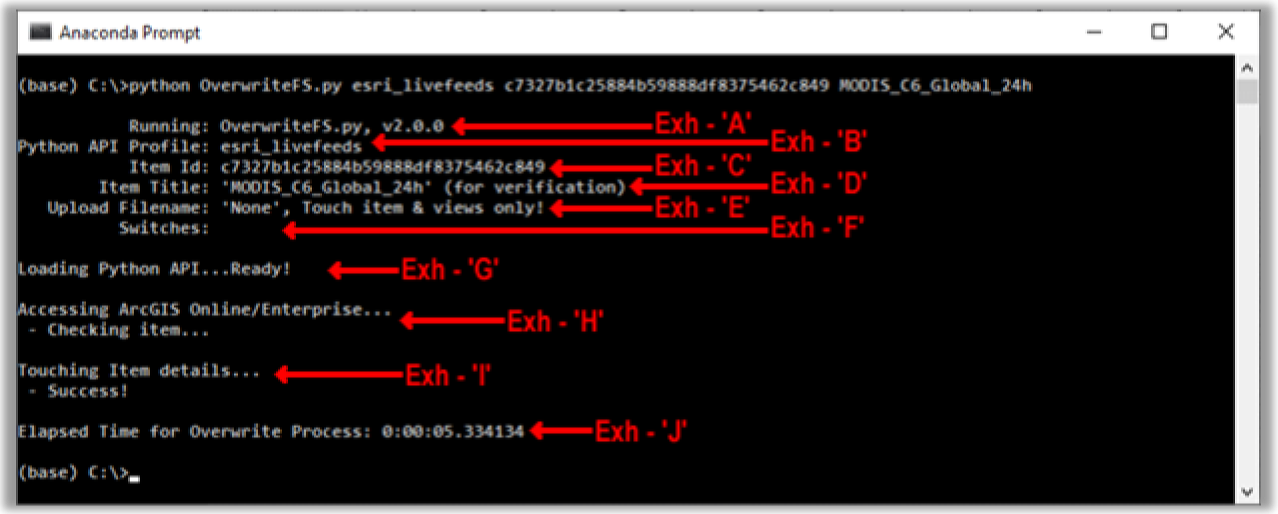
- Execution with optional help Switch '*-h*' specified


Anaconda Prompt

```
(base) C:\>python OverwriteFS.py -h

v2.0.0 Usage: Python OverwriteFS.py [-h] <profile> <item> <title> [<filename> | <url>] [-OutPath <output folder>] [-NoTi
meSeries] [-NoIndexes] [-NoTouch] [-NoWait] [-NoProps | -PersistProps] [-DryRun] [-GetTarget | -SwapLayers | [-ListRelat
ed] [-AddRelated | -RemoveRelated [<Item A id>[ <Item B id>]]]] [-Convert <module>[ <call param>[ <call param>[ ...]]]]
[-AllowPWprompt] [-LessDetail | -MoreDetail] [-Password <password>]

           -h: (optional) Action Switch that triggers 'usage' display and exit.

    <profile>: (required) Stored Python API user Profile to connect with.
                          Specify 'Pro' to leverage active ArcGIS Pro connection, also requires Arcpy!

       <item>: (required) Item Id of Feature Service or Feature View to act on.

      <title>: (required) Title of <item>, to verify item Id matches correct item.

   <filename>: (optional) File path/name, or URL containing Service data to Overwrite with.
      or <url>            If 'path' only, script will search path for original filename used to publish Service.
                          Default: Empty string, only touch Item and/or Views in order to refresh last update date.

     -OutPath: (optional) Option Switch setting output file Path used to store <url> download data during run.
                          * Note * Ignored when specified <filename> is used!
                          Default: User's Temporary folder.

 -NoTimeSeries: (optional) Option Switch instructing function not to touch Time Extent of Time Series enabled Layers
                          in Service and related Views.
                          Default: Touch Layer Time Information of Item and related Views, reflecting new data.

   -NoIndexes: (optional) Option Switch instructing function not to recreate missing Layer Indexes on Service.
                          Default: Recreate indexes if they are missing after an Overwrite action.

     -NoTouch: (optional) Option Switch instructing function not to update the 'lastUpdated' item property for the
                          Service or related Views unless an update has been made to them.
                          Default: Update, or 'touch', the 'lastUpdate' property on the Service item and Views, even
                                   when an update has not been made. Signifies that the process has been run.

      -NoWait: (optional) Option Switch instructing function not to wait for re-application of properties like Layer
                          Optimization to complete before continuing the Overwrite or Layer Swap action. When enabled,
                          function will report condition and supply a URL that can be used for manual status review.
                          Default: Function will wait for properties like Layer Optimization to be re-applied before
                                   proceeding to the next processing 'step' in the Overwrite or Layer Swap workflow.

     -NoProps: (optional) Option Switch instructing function NOT to Re-Apply Service or View properties following
                          a successful update. This Defaults Service or View back to its Published state!
                          Default: Re-Apply properties and do not persist Backup file beyond successful update.

 -PersistProps: (optional) Option Switch instructing function to retain Service or View property Backup File
                          after a successful Overwrite and property restoration.
                          Default: Re-Apply properties and do not persist Backup file beyond successful update.

      -DryRun: (optional) Option Switch instructing function to step through process WITHOUT updating the Service.
                          Default: Update or Touch the Service and Item.

  -SwapLayers: (optional) Action Switch instructing function to Swap Layers in View, point all Layers to Target
                          or newly Overwritten Feature Service. Used by A/B Feature Service enabled View, whereby
                          the View is Related to Two Feature Services, allowing the View's Layers to be swapped
                          out to point to the matching Layers of the newly updated Feature Service.
                          Default: Overwrite action when <filename> specified and no other Action Switches included.

   -GetTarget: (optional) Action Switch instructing function to report the selected Feature Service item Id, Title,
                          and File Item upload Filename that should be used for the next update target.
                          Leveraged by A/B View enabled Services to select inactive
                          Feature Service that should be overwritten next.
                          Default: Overwrite action when <filename> specified and no other switches included.

 -ListRelated: (optional) Action Switch instructing function to List all Items related to <item> View or Service
                          Default: Overwrite action when <filename> specified and no other switches included.

  -AddRelated: (optional) Action Switch instructing function to Add specified Service Items to View <item>, limit two
                          item Ids as Related Services. Related Services are required for 'SwapLayers' action to
                          switch a View's Layers so they point to Related 'target' Feature Service Layers.
                          Default: If No Items specified, action will be to 'ListRelated' <item>'s Relationships.

-RemoveRelated: (optional) Action Switch instructing function to Remove specified A/B Service Items from View <item>.
                          Related Feature Services are required for 'SwapLayers' to target and switch Layers of
                          <item> View so they point to same Layers referenced by Related 'target' Feature Service.
                          Default: If No Items specified, action will be to 'ListRelated' <item>'s Relationships.

  <Item A id>: (optional, but at least one is required when using -AddRelated or -RemoveRelated Action Switch)
or <Item B id>            The unique Item Identifier for the Content Item managing the Service.
                          Ex: Id for 'Active Hurricanes, Cyclones, and Typhoons' is: 248e7b5827a34b248647afb012c58787

     -Convert: (optional) Action Switch instructing function to import specified Python Module and use it to
                          transform the source data before Overwriting the Hosted Feature Service.
                          * Note * Only available during a Overwrite action.
                          Default: No conversion of the input data will take place.

     <module>: (optional, but required when using -Convert Action Switch) A Case-Sensitive Python Module (or script)
                          name to leverage for data conversion prior to updating Service. Supports Python 'Dot'
                          notation for Folder/Module/Class path access to 'convert' Function.
                          * Note * Path is RELATIVE to 'Converters' folder in OverwriteFS.py script location!
                          Ex: <folder>.<module>.<class>

  <call param>: (optional) Additional call parameters to pass to '-Convert' Module function. Only need to include
                          parameters that follow the download file specifier in 'convert' Function.

-AllowPWprompt: (optional) Option Switch instructing function to Allow use of an undefined user account password
                          in the Profile, allowing Python API to prompt for user entry.
                          * Note * Not recommended for use during Unattended execution!
                          Default: If Profile Password is not defined, function will report the condition and Exit.

  -LessDetail: (optional) Option Switch instructing function to only Display major steps and error responses.
                          Default: Display step by step processing detail, or -MoreDetail if set.

  -MoreDetail: (optional) Option Switch instructing function to Display maximum Diagnostic detail and error responses.
                          Default: Display step by step processing detail, or -LessDetail if set.

    -Password: (optional) Option Parameter instructing function to overwride the stored Profile password with this
                          plain text password.
                          Default: Use password set in Profile.
   <password>: (optional, but required when using -Password Option)

(base) C:\>_
```

- Identifying Response Details reported during execution

When executing script, identify specific details reported during the run. See following figure:



```
(base) C:\>python OverwriteFS.py esri_livefeeds c7327b1c25884b59888df8375462c849 MODIS_C6_Global_24h

           Running: OverwriteFS.py, v2.0.0
Python API Profile: esri_livefeeds
           Item Id: c7327b1c25884b59888df8375462c849
        Item Title: 'MODIS_C6_Global_24h' (for verification)
   Upload Filename: 'None', Touch item & views only!
          Switches:

Loading Python API...Ready!

Accessing ArcGIS Online/Enterprise...
 - Checking item...

Touching Item details...
 - Success!

Elapsed Time for Overwrite Process: 0:00:05.334134

(base) C:\>_
```

Exhibit – ‘A’: Reports the filename and version of the script that was launched.

Exhibit – ‘B’: Reports the Python API Profile specified at launch time.

Exhibit – ‘C’: Reports the Item Id of the Item the script was instructed to act on.

Exhibit – ‘D’: Reports the Item Title provided, as verification that the script will act on the desired Item.

Exhibit – ‘E’: Reports the update file provided. In this case, a file was not provided, so the script reports that it will only “touch” the item. If a filename, folder name, or URL were provided, the script would have reported the appropriate details for the record.

Exhibit – ‘F’: Reports any Option Switches that were provided.

Exhibit – ‘G’: Reports the progress of the script during importation of the Python API.

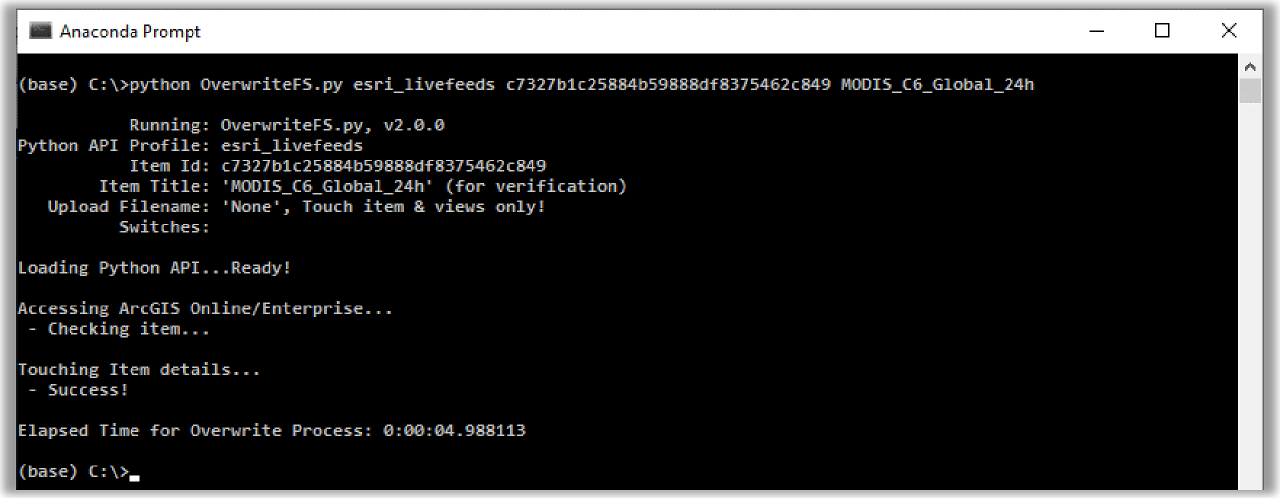
Exhibit – ‘H’: Reports the progress of the script using the Python API to connect to the Enterprise or Online ArcGIS Portal.

Exhibit – ‘I’: Reports the progress of the desired action, Item “touch” in this case.

Exhibit – ‘J’: Reports the Elapsed processing time for the run in <hours>:<minutes>:<seconds>

- Touching a Service

Without providing a file name or URL path to update from, the script will “touch” the Service, Service Item, and related Views. No data changes will be made, but this action reflects that the update process has run, updating the ‘lastUpdateDate’ property for all Items.

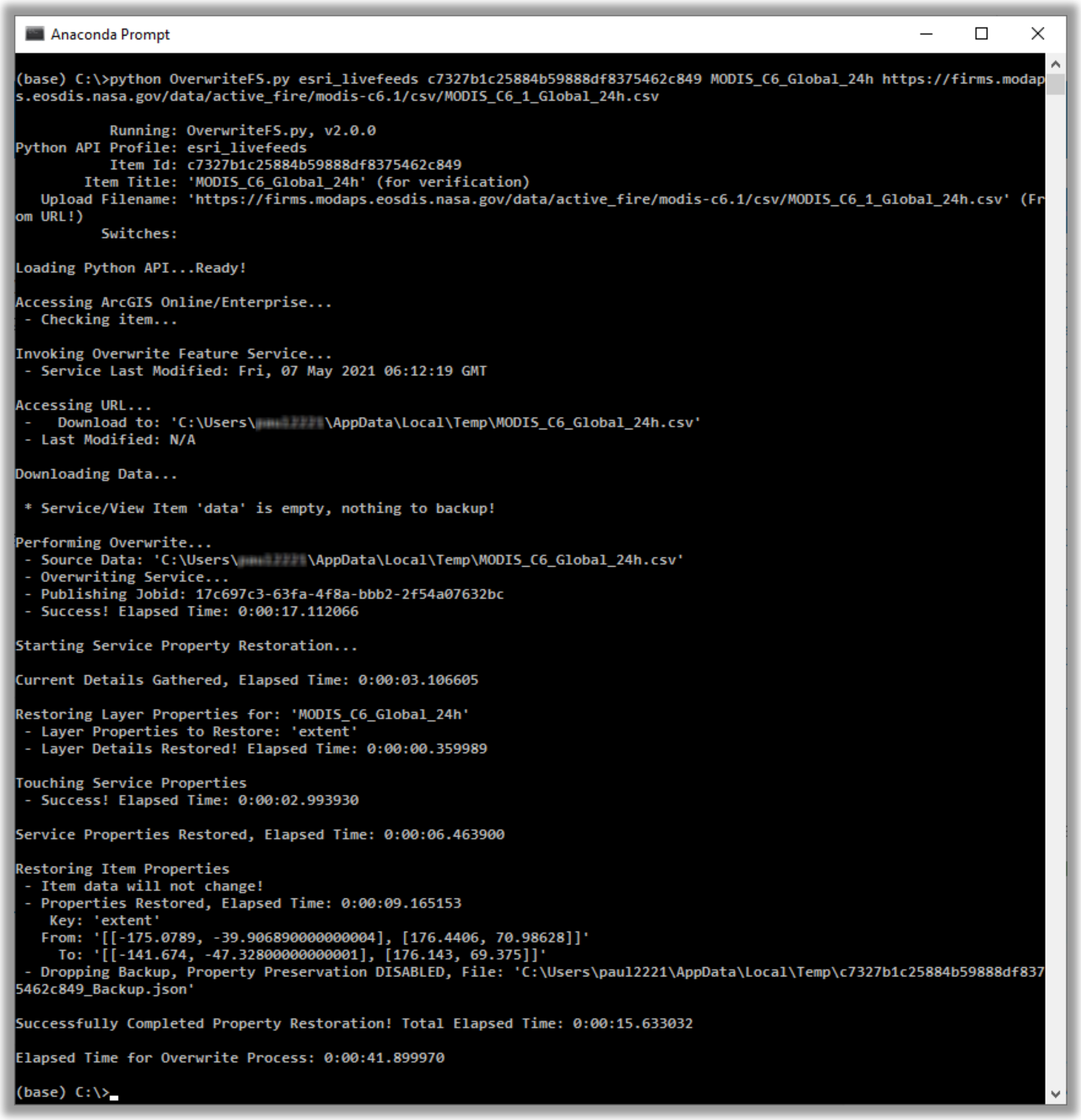

Anaconda Prompt

```
(base) C:\>python OverwriteFS.py esri_livefeeds c7327b1c25884b59888df8375462c849 MODIS_C6_Global_24h

           Running: OverwriteFS.py, v2.0.0
Python API Profile: esri_livefeeds
           Item Id: c7327b1c25884b59888df8375462c849
        Item Title: 'MODIS_C6_Global_24h' (for verification)
   Upload Filename: 'None', Touch item & views only!
          Switches:

Loading Python API...Ready!

Accessing ArcGIS Online/Enterprise...
 - Checking item...

Touching Item details...
 - Success!

Elapsed Time for Overwrite Process: 0:00:04.988113

(base) C:\>
```

- Overwriting a Service, renewing its Data

Overwriting an existing Service is the act of Re-Publishing from the updated data source content. This action will recreate the Service and Service Item as if it were newly published, reverting all Post-Publishing property changes back to their original state. Prior to performing the Overwrite, the script will backup existing properties and try to restore them following a successful Overwrite action. In the event the Overwrite action or Property restoration is unsuccessful, the Backup file will be retained.

The following is an example update of the MODIS past 24h Active Fire Data Service with current CSV formatted data maintained by NASA.



Anaconda Prompt

```
(base) C:\>python OverwriteFS.py esri_livefeeds c7327b1c25884b59888df8375462c849 MODIS_C6_Global_24h https://firms.modap
s.eosdis.nasa.gov/data/active_fire/modis-c6.1/csv/MODIS_C6_1_Global_24h.csv

           Running: OverwriteFS.py, v2.0.0
Python API Profile: esri_livefeeds
           Item Id: c7327b1c25884b59888df8375462c849
        Item Title: 'MODIS_C6_Global_24h' (for verification)
   Upload Filename: 'https://firms.modaps.eosdis.nasa.gov/data/active_fire/modis-c6.1/csv/MODIS_C6_1_Global_24h.csv' (Fr
om URL!)
          Switches:

Loading Python API...Ready!

Accessing ArcGIS Online/Enterprise...
 - Checking item...

Invoking Overwrite Feature Service...
 - Service Last Modified: Fri, 07 May 2021 06:12:19 GMT

Accessing URL...
 -   Download to: 'C:\Users\[redacted]\AppData\Local\Temp\MODIS_C6_Global_24h.csv'
 - Last Modified: N/A

Downloading Data...

 * Service/View Item 'data' is empty, nothing to backup!

Performing Overwrite...
 - Source Data: 'C:\Users\[redacted]\AppData\Local\Temp\MODIS_C6_Global_24h.csv'
 - Overwriting Service...
 - Publishing Jobid: 17c697c3-63fa-4f8a-bbb2-2f54a07632bc
 - Success! Elapsed Time: 0:00:17.112066

Starting Service Property Restoration...

Current Details Gathered, Elapsed Time: 0:00:03.106605

Restoring Layer Properties for: 'MODIS_C6_Global_24h'
 - Layer Properties to Restore: 'extent'
 - Layer Details Restored! Elapsed Time: 0:00:00.359989

Touching Service Properties
 - Success! Elapsed Time: 0:00:02.993930

Service Properties Restored, Elapsed Time: 0:00:06.463900

Restoring Item Properties
 - Item data will not change!
 - Properties Restored, Elapsed Time: 0:00:09.165153
    Key: 'extent'
   From: '[[-175.0789, -39.906890000000004], [176.4406, 70.98628]]'
     To: '[[-141.674, -47.328000000000001], [176.143, 69.375]]'
 - Dropping Backup, Property Preservation DISABLED, File: 'C:\Users\paul2221\AppData\Local\Temp\c7327b1c25884b59888df837
5462c849_Backup.json'

Successfully Completed Property Restoration! Total Elapsed Time: 0:00:15.633032

Elapsed Time for Overwrite Process: 0:00:41.899970

(base) C:\>
```

- Invoking the '***-Convert***' option

In some cases, you may need or want to alter, cleanup, or augment the data you are accessing from the source **prior** to updating your Service. To help with this, the OverwriteFS script includes a '***-Convert***' option that allows you to specify a custom Python script that can process the incoming data file and return an updated/altered data file.

The only requirement for this custom script is that it includes a Function called 'convert', and this Function accepts a text string parameter that will contain the file location and name of the source data file that was downloaded or specified for updating the Service. The convert Function is only required to return a text string containing the new or updated file path and name OR False/None, indicating that there is no Service update required (because the source data has not changed). You can include optional Python parameters as needed. The OverwriteFS script will check for a 'verbose' parameter, which will be populated with the Less/More Details (verbosity) Boolean flag, allowing user to include or exclude displaying feedback from the script as desired.

As an example of this workflow, the install package includes a conversion script called '**Rss2Json**'. This script reads an XML based RSS or GeoRSS file (not currently supported by Online or Portal) and creates a GeoJson file based on the Tags and attributes found within it. The output can then be used to update your Service. **See the script-specific documentation for more details and examples!**

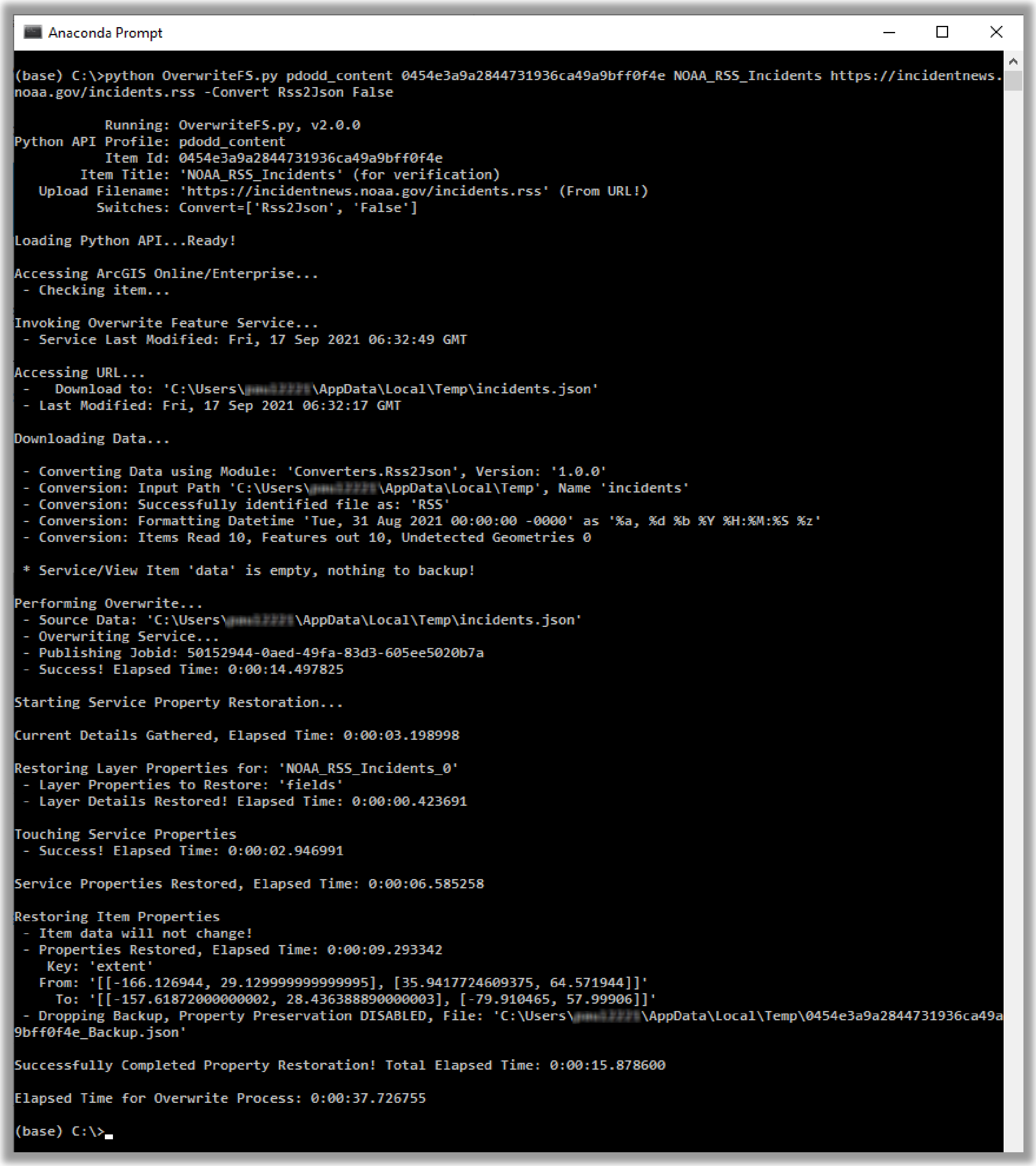
*** Tip *** Download the source data manually first and then pass it into your custom script. The output file can then be used to tailor the processing and initially create your Service!

*** Tip *** The Rss2Json script will generate a source-filename specific INI file alongside the source file. This INI file contains field order, naming, and other field specific details that can help customize, configure, or control the output. **See Converter script documentation for more details!**

The following are example conversion runs that read NOAA's Incident RSS NEWS channel to update a Hosted Feature Service with current Geo Spatial Incidents. Notice the 'Convert' option and output ' - Conversion:' detail between the 'Download' and 'Performing Overwrite' steps.

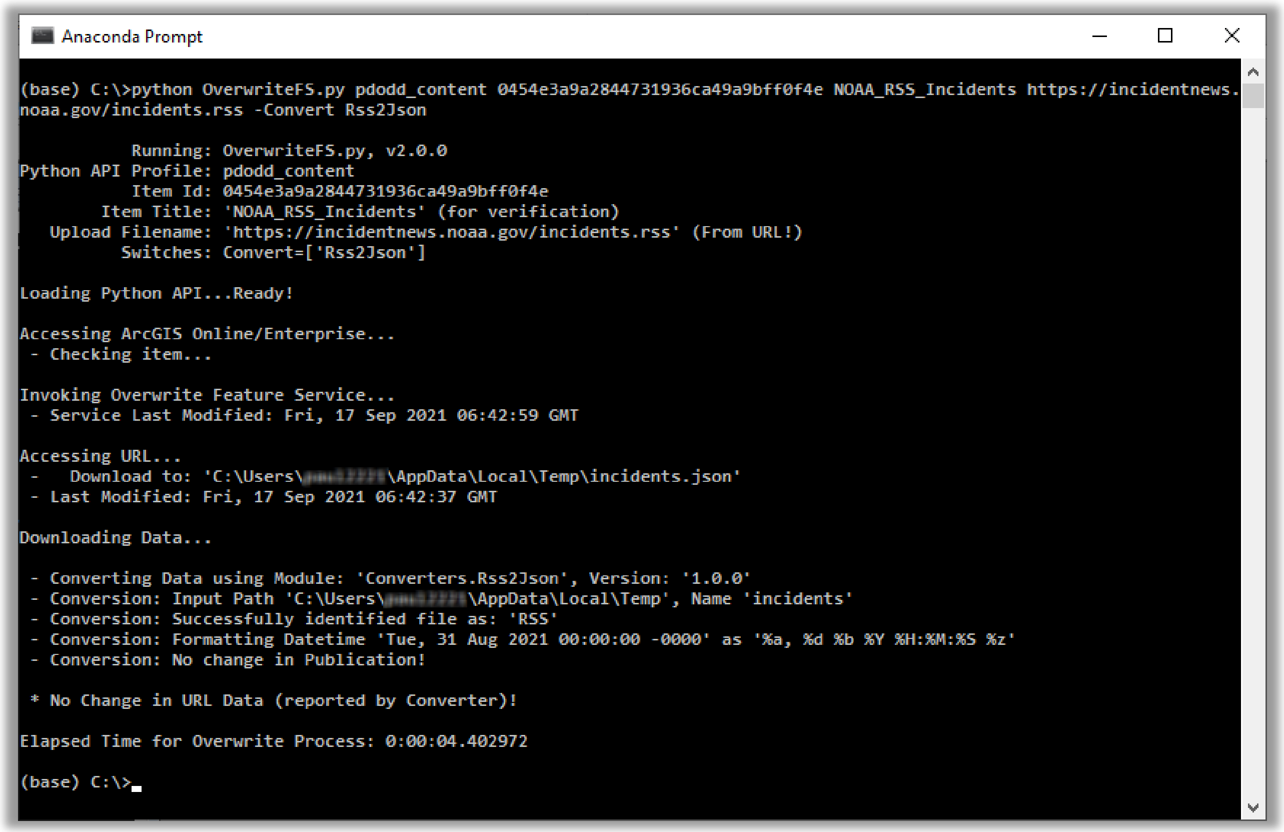
The Rss2Json converter script accepts one additional parameter, a True or False optional property that instructs the routine to compare or ignore the RSS Publication date. The output of the first run below includes a 'False' setting for this property, allowing the routine to run without comparing the publication date.

The second run compares the publication date and determines that the publication date has not changed since the last run, so it returns False, or no conversion file. This tells the OverwriteFS script that there is no need to perform the overwrite action because the data has not changed.



```
(base) C:\>python OverwriteFS.py pdodd_content 0454e3a9a2844731936ca49a9bff0f4e NOAA_RSS_Incidents https://incidentnews.
noaa.gov/incidents.rss -Convert Rss2Json False

           Running: OverwriteFS.py, v2.0.0
Python API Profile: pdodd_content
           Item Id: 0454e3a9a2844731936ca49a9bff0f4e
        Item Title: 'NOAA_RSS_Incidents' (for verification)
   Upload Filename: 'https://incidentnews.noaa.gov/incidents.rss' (From URL!)
          Switches: Convert=['Rss2Json', 'False']

Loading Python API...Ready!

Accessing ArcGIS Online/Enterprise...
 - Checking item...

Invoking Overwrite Feature Service...
 - Service Last Modified: Fri, 17 Sep 2021 06:32:49 GMT

Accessing URL...
 -   Download to: 'C:\Users\[redacted]\AppData\Local\Temp\incidents.json'
 - Last Modified: Fri, 17 Sep 2021 06:32:17 GMT

Downloading Data...

 - Converting Data using Module: 'Converters.Rss2Json', Version: '1.0.0'
 - Conversion: Input Path 'C:\Users\[redacted]\AppData\Local\Temp', Name 'incidents'
 - Conversion: Successfully identified file as: 'RSS'
 - Conversion: Formatting Datetime 'Tue, 31 Aug 2021 00:00:00 -0000' as '%a, %d %b %Y %H:%M:%S %z'
 - Conversion: Items Read 10, Features out 10, Undetected Geometries 0

 * Service/View Item 'data' is empty, nothing to backup!

Performing Overwrite...
 - Source Data: 'C:\Users\[redacted]\AppData\Local\Temp\incidents.json'
 - Overwriting Service...
 - Publishing Jobid: 50152944-0aed-49fa-83d3-605ee5020b7a
 - Success! Elapsed Time: 0:00:14.497825

Starting Service Property Restoration...

Current Details Gathered, Elapsed Time: 0:00:03.198998

Restoring Layer Properties for: 'NOAA_RSS_Incidents_0'
 - Layer Properties to Restore: 'fields'
 - Layer Details Restored! Elapsed Time: 0:00:00.423691

Touching Service Properties
 - Success! Elapsed Time: 0:00:02.946991

Service Properties Restored, Elapsed Time: 0:00:06.585258

Restoring Item Properties
 - Item data will not change!
 - Properties Restored, Elapsed Time: 0:00:09.293342
    Key: 'extent'
   From: '[[-166.126944, 29.129999999999995], [35.9417724609375, 64.571944]]'
     To: '[[-157.61872000000002, 28.436388890000003], [-79.910465, 57.99906]]'
 - Dropping Backup, Property Preservation DISABLED, File: 'C:\Users\[redacted]\AppData\Local\Temp\0454e3a9a2844731936ca49a
9bff0f4e_Backup.json'

Successfully Completed Property Restoration! Total Elapsed Time: 0:00:15.878600

Elapsed Time for Overwrite Process: 0:00:37.726755

(base) C:\>_
```

Output from 'Convert Rss2Json' Run #1



```
(base) C:\>python OverwriteFS.py pdodd_content 0454e3a9a2844731936ca49a9bff0f4e NOAA_RSS_Incidents https://incidentnews.
noaa.gov/incidents.rss -Convert Rss2Json

           Running: OverwriteFS.py, v2.0.0
Python API Profile: pdodd_content
           Item Id: 0454e3a9a2844731936ca49a9bff0f4e
        Item Title: 'NOAA_RSS_Incidents' (for verification)
   Upload Filename: 'https://incidentnews.noaa.gov/incidents.rss' (From URL!)
          Switches: Convert=['Rss2Json']

Loading Python API...Ready!

Accessing ArcGIS Online/Enterprise...
 - Checking item...

Invoking Overwrite Feature Service...
 - Service Last Modified: Fri, 17 Sep 2021 06:42:59 GMT

Accessing URL...
 -   Download to: 'C:\Users\[redacted]\AppData\Local\Temp\incidents.json'
 - Last Modified: Fri, 17 Sep 2021 06:42:37 GMT

Downloading Data...

 - Converting Data using Module: 'Converters.Rss2Json', Version: '1.0.0'
 - Conversion: Input Path 'C:\Users\[redacted]\AppData\Local\Temp', Name 'incidents'
 - Conversion: Successfully identified file as: 'RSS'
 - Conversion: Formatting Datetime 'Tue, 31 Aug 2021 00:00:00 -0000' as '%a, %d %b %Y %H:%M:%S %z'
 - Conversion: No change in Publication!

 * No Change in URL Data (reported by Converter)!

Elapsed Time for Overwrite Process: 0:00:04.402972

(base) C:\>
```

Output from 'Convert Rss2Json' Run #2

*** Note * See the Rss2Json documentation for additional details, concerns, and options when using this script!**

- Repairing a lost Service to File Item relationship

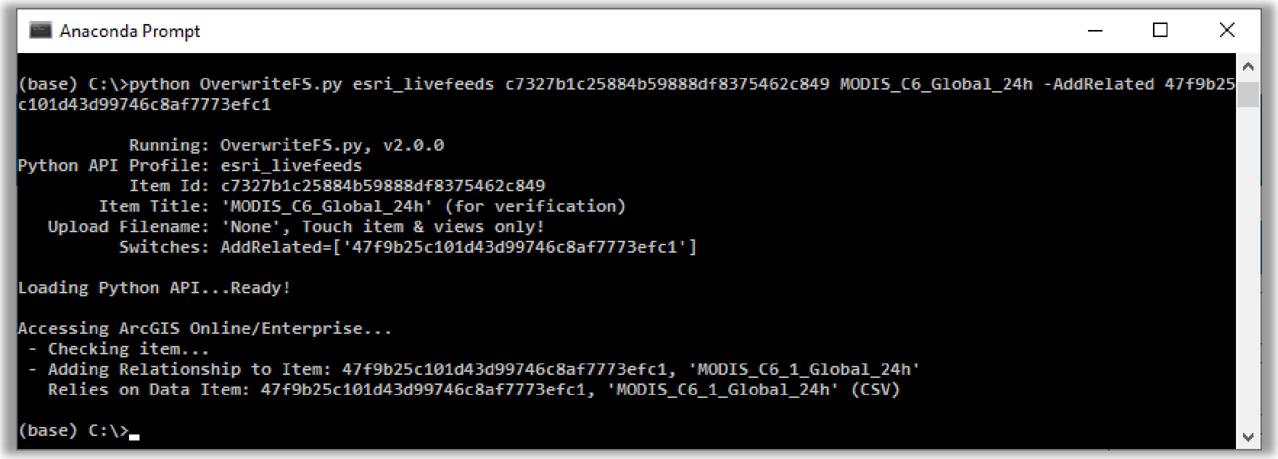
Content Items in Online or Portals often relate to one another, supporting stored relationships. For example, Hosted Feature Services can have related Views that offer a selective rendering or obfuscation of the data, fields, or service extent. The View Item is considered a Child to the Service Parent.

Another relationship is formed when data is Published with the intent of creating a Service. The Publish action creates a File Data Item that stores a copy of the Service data PRIOR to creating the Service. This 'File' Item can contain GeoJSON, CSV, Excel, or SD (Service Definition) files. Once a File Item has been uploaded, you can elect to publish a Service from this Item after upload or during the Publish action. This Publishing action creates a relationship between the File Item and Hosted Service Item. The Service is a Child to the File Item in this case. Overwriting a Service to update the data or schema REQUIRES that this File Item to Feature Service Item relationship exists. If for some reason the File Item for a Service is removed or deleted, you will not be able to Overwrite the Service!

To repair this relationship, the OverwriteFS script can relate a Service Item back to a File Item, restoring the ability to Overwrite the Service.

Restoration Steps:

1) Confirm that the appropriate File Item does not already exist, upload a copy of the file item using the Online or Portal User Interface if not. **DO NOT create a Service item during the upload!** You will not be able to relate this File Item to the existing Service if you create a new Service. A File Item can only relate to one Service at a time. When specifying the Title of the File Item, match this to the Service Item Title. Otherwise the next Service Overwrite action will alter the Service Item Title to match the File Item Title.
2) Open a Python Command prompt.
3) Launch Python and include the OverwriteFS script, specifying the Profile that owns the Service item, the Service Item Id, Service Title, '***-AddRelated***' (Action Switch to add a relationship), followed by the Item Id of the File Item that should be related to the Service. You always want to update the Child Item with its related Parent(s). The Service is a Child to the File Item, so we want to alter the Service Item so it will relate back to the newly uploaded Parent File Item! See screen shot below that relates the MODIS Service Item to a new File Item after the original File Item was accidently deleted.


Anaconda Prompt

```
(base) C:\>python OverwriteFS.py esri_livefeeds c7327b1c25884b59888df8375462c849 MODIS_C6_Global_24h -AddRelated 47f9b25
c101d43d99746c8af7773efc1

           Running: OverwriteFS.py, v2.0.0
Python API Profile: esri_livefeeds
           Item Id: c7327b1c25884b59888df8375462c849
        Item Title: 'MODIS_C6_Global_24h' (for verification)
   Upload Filename: 'None', Touch item & views only!
          Switches: AddRelated=['47f9b25c101d43d99746c8af7773efc1']

Loading Python API...Ready!

Accessing ArcGIS Online/Enterprise...
 - Checking item...
 - Adding Relationship to Item: 47f9b25c101d43d99746c8af7773efc1, 'MODIS_C6_1_Global_24h'
   Relies on Data Item: 47f9b25c101d43d99746c8af7773efc1, 'MODIS_C6_1_Global_24h' (CSV)

(base) C:\>
```

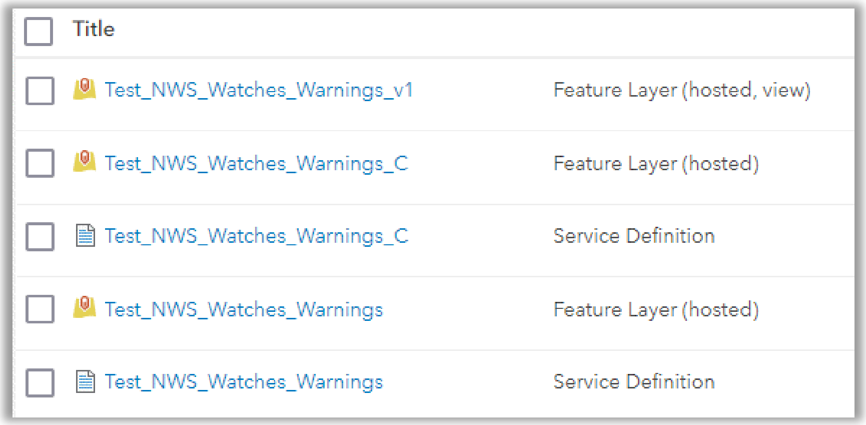
- Adding Multi-Service Relationships to a View

In some workflows, having more than one Parent Item related to a Child may be required. The Swap Layers workflow for instance is one that requires a Child View to have two related Hosted Feature Service Parents, so that the OverwriteFS script will know which Services can participate in the Swap.

To accomplish this, we will leverage the '*-AddRelated*' and '*-ListRelated*' Action Switches. Then we will verify the relationship configuration by using the '*-GetTarget*' Action Switch to see what Service Item will be targeted by the Swap Layers process later on.

Consider first that have a Hosted Feature Service with a Child View created. To support the Swap Layers workflow, we need to publish a second Hosted Feature Service so we have multiple target Services that the Swap operation can interact with. Once the new Service has been published, we can add the Parent/Child relationship to the View!

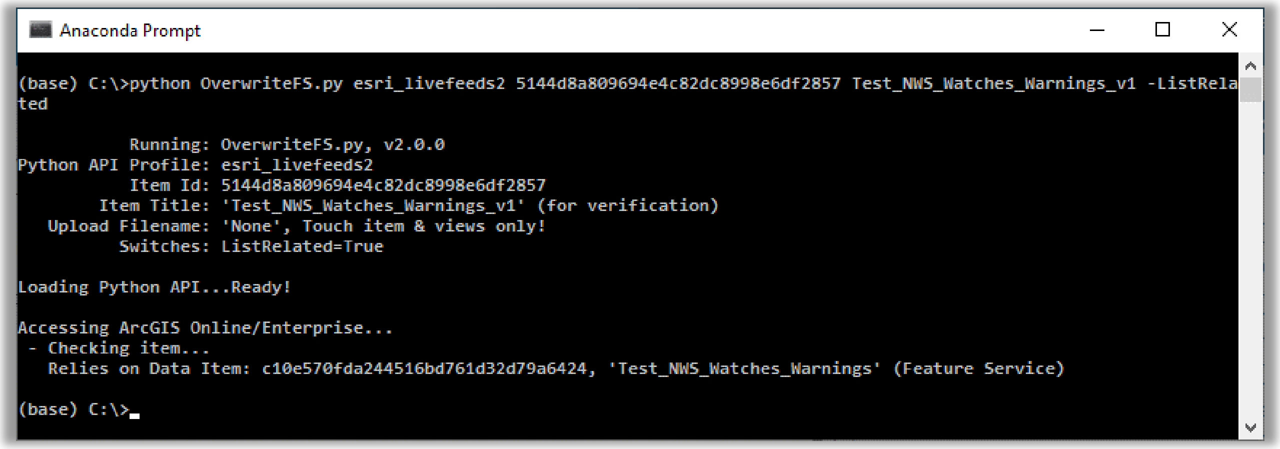
Here is an Item list showing a pair of Test '*Watches, Warnings, and Advisory*' Hosted Feature Services with their File Items, and a Hosted Feature View built from the original Service, not ending in '_C':



Title	
Test_NWS_Watches_Warnings_v1	Feature Layer (hosted, view)
Test_NWS_Watches_Warnings_C	Feature Layer (hosted)
Test_NWS_Watches_Warnings_C	Service Definition
Test_NWS_Watches_Warnings	Feature Layer (hosted)
Test_NWS_Watches_Warnings	Service Definition

Using the '*-ListRelated*' Action Switch, we can list the related Items on the View, showing us that it only has one relationship to the original Hosted Feature Service: '*Test_NWS_Watches_Warnings*'

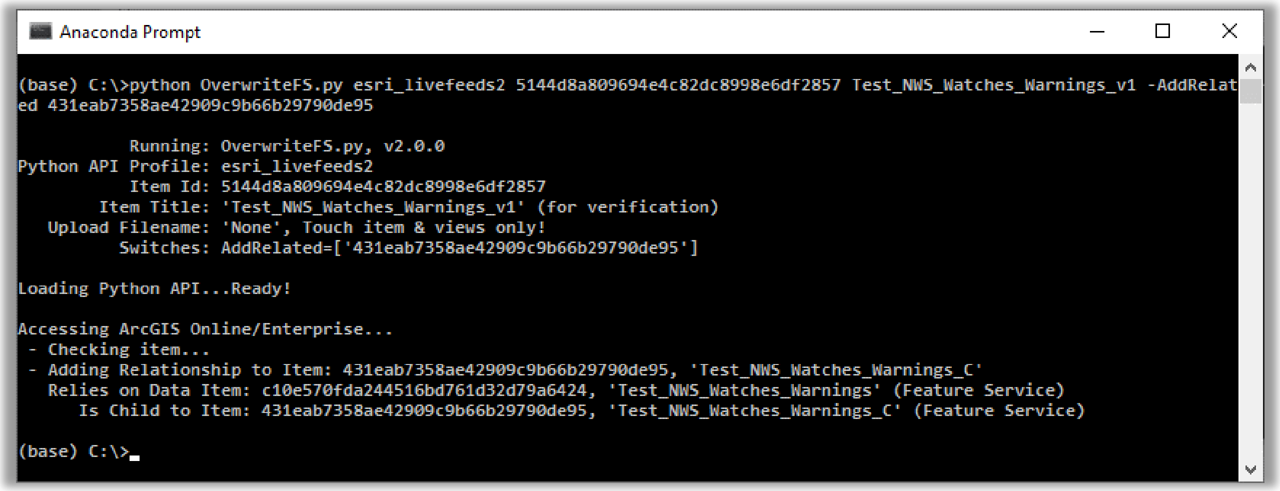
The Child View is said to 'rely' on a Parent Service, as a source Data Item in this case.



```
(base) C:\>python OverwriteFS.py esri_livefeeds2 5144d8a809694e4c82dc8998e6df2857 Test_NWS_Watches_Warnings_v1 -ListRela
ted

           Running: OverwriteFS.py, v2.0.0
Python API Profile: esri_livefeeds2
           Item Id: 5144d8a809694e4c82dc8998e6df2857
        Item Title: 'Test_NWS_Watches_Warnings_v1' (for verification)
   Upload Filename: 'None', Touch item & views only!
          Switches: ListRelated=True

Loading Python API...Ready!

Accessing ArcGIS Online/Enterprise...
 - Checking item...
   Relies on Data Item: c10e570fda244516bd761d32d79a6424, 'Test_NWS_Watches_Warnings' (Feature Service)

(base) C:\>_
```

Using the '*-AddRelated*' Action Switch, we can relate a second Service to the View, adding the '*Test_NWS_Watches_Warnings_C*' Hosted Feature Service to the mix.

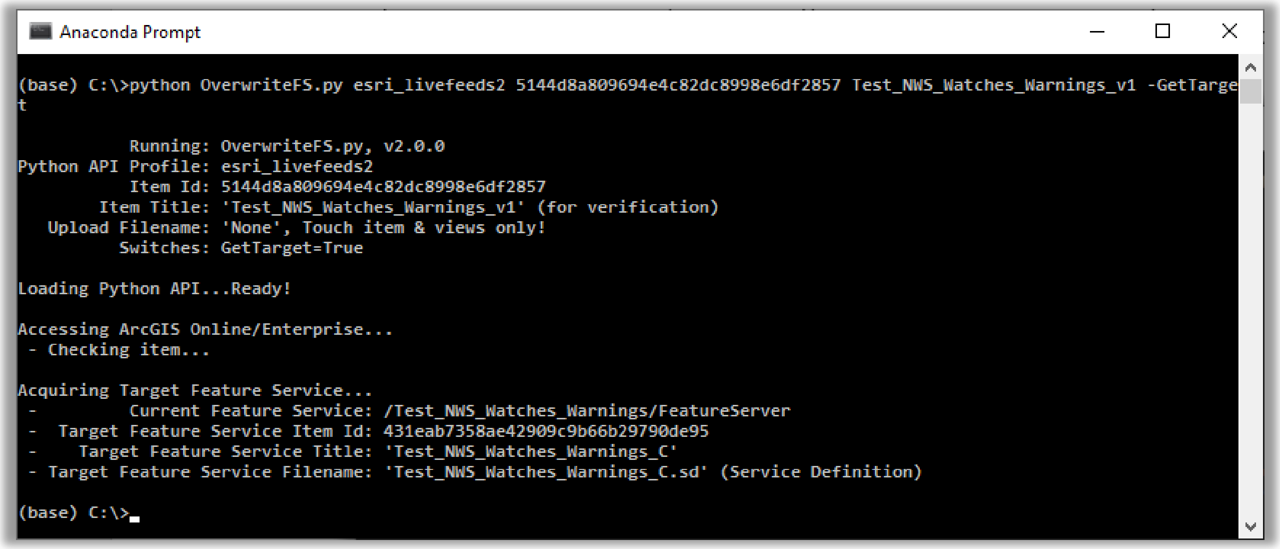
Now you can see that the View is a Child to the second Service. This is just an idle relationship, compared to the relationship to the original Service. Because the Layers in the View are pointing to the Layers in the original Service, so the View 'relies' on the original Service for its data!



```
(base) C:\>python OverwriteFS.py esri_livefeeds2 5144d8a809694e4c82dc8998e6df2857 Test_NWS_Watches_Warnings_v1 -AddRelat
ed 431eab7358ae42909c9b66b29790de95

           Running: OverwriteFS.py, v2.0.0
Python API Profile: esri_livefeeds2
           Item Id: 5144d8a809694e4c82dc8998e6df2857
        Item Title: 'Test_NWS_Watches_Warnings_v1' (for verification)
   Upload Filename: 'None', Touch item & views only!
          Switches: AddRelated=['431eab7358ae42909c9b66b29790de95']

Loading Python API...Ready!

Accessing ArcGIS Online/Enterprise...
 - Checking item...
 - Adding Relationship to Item: 431eab7358ae42909c9b66b29790de95, 'Test_NWS_Watches_Warnings_C'
   Relies on Data Item: c10e570fda244516bd761d32d79a6424, 'Test_NWS_Watches_Warnings' (Feature Service)
      Is Child to Item: 431eab7358ae42909c9b66b29790de95, 'Test_NWS_Watches_Warnings_C' (Feature Service)

(base) C:\>
```

With the View properly configured to support two Parent Hosted Feature Service Items, we can check the configuration to preview what Service Item the Swap Layers process will select.

From the output of running with the '*-GetTarget*' Action Switch, we can see that the script has identified the original Hosted Feature Service as the 'Current' source for the View, and it has identified the newly added Service as the potential 'Target' source. ***Success, we are ready to support a Swap!***



```
(base) C:\>python OverwriteFS.py esri_livefeeds2 5144d8a809694e4c82dc8998e6df2857 Test_NWS_Watches_Warnings_v1 -GetTarge
t

           Running: OverwriteFS.py, v2.0.0
Python API Profile: esri_livefeeds2
           Item Id: 5144d8a809694e4c82dc8998e6df2857
        Item Title: 'Test_NWS_Watches_Warnings_v1' (for verification)
   Upload Filename: 'None', Touch item & views only!
          Switches: GetTarget=True

Loading Python API...Ready!

Accessing ArcGIS Online/Enterprise...
 - Checking item...

Acquiring Target Feature Service...
 -           Current Feature Service: /Test_NWS_Watches_Warnings/FeatureServer
 -   Target Feature Service Item Id: 431eab7358ae42909c9b66b29790de95
 -     Target Feature Service Title: 'Test_NWS_Watches_Warnings_C'
 - Target Feature Service Filename: 'Test_NWS_Watches_Warnings_C.sd' (Service Definition)

(base) C:\>
```

- Relationship Direction

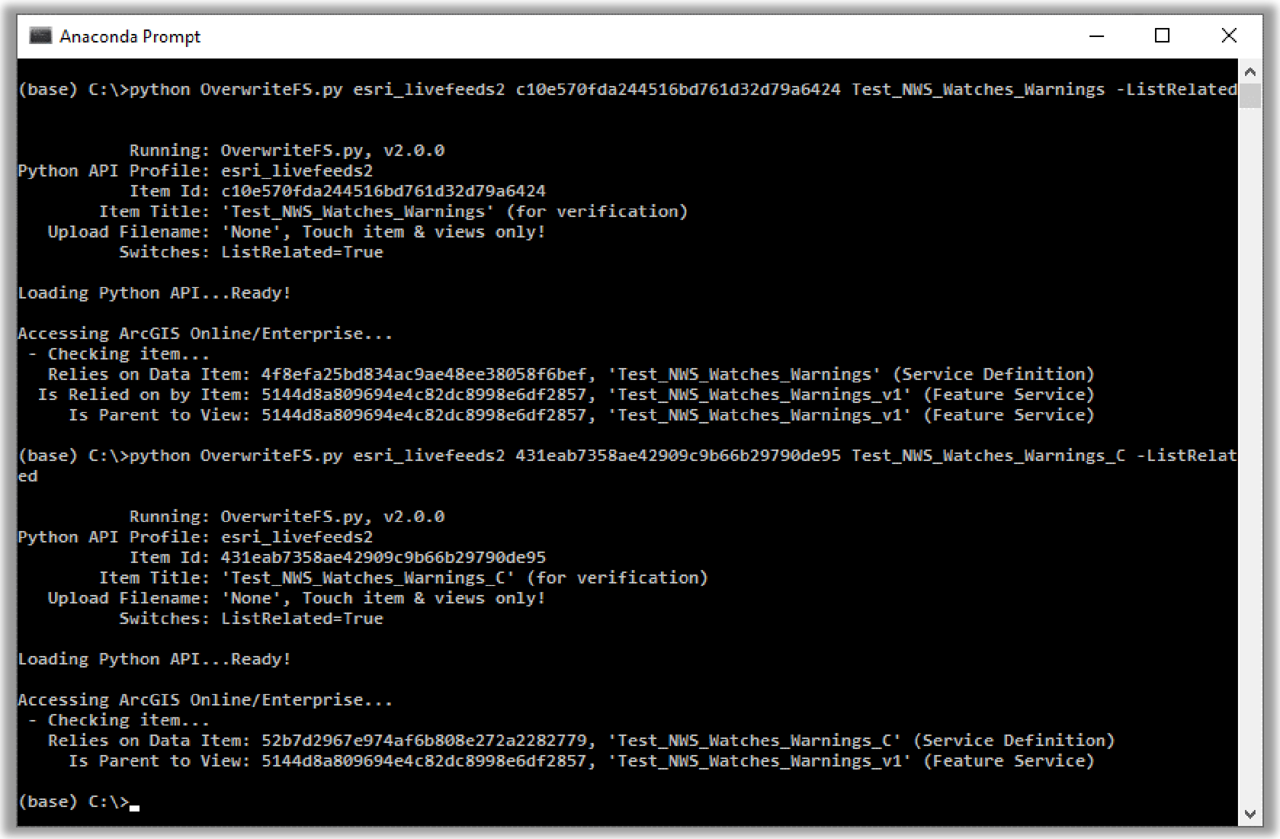
So far, we have only reviewed relationships from the Child Item perspective. What does this look like from the Parent Item Perspective?

If we run the script, listing the relationships for both Hosted Feature Services, we should see a more complete picture of how Items relate to one another.

In the following screenshot, we can see that each Service Item 'relies' on, is a Child to, its Parent Service Definition 'File' Item. This is a Data relationship. We can also see that each Service Item, is a Parent to the Child View Item. This is a Service-to-Service relationship. And, we can tell that the Child View Item is dependent on the Original Parent Service Item. Also, a Data relationship.

So, depending on what Item you are looking at, the Parent to Child or Child to Parent relationship can be discovered and recognized by the relationship terminology:

- Parent to Child: 'Is Relied on', 'Is Parent to'
- Child to Parent: 'Relies on', 'Is Child to'



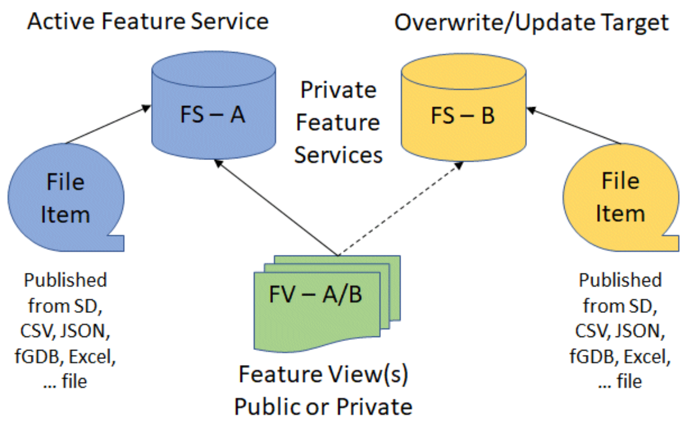
```
(base) C:\>python OverwriteFS.py esri_livefeeds2 c10e570fda244516bd761d32d79a6424 Test_NWS_Watches_Warnings -ListRelated


           Running: OverwriteFS.py, v2.0.0
Python API Profile: esri_livefeeds2
           Item Id: c10e570fda244516bd761d32d79a6424
        Item Title: 'Test_NWS_Watches_Warnings' (for verification)
   Upload Filename: 'None', Touch item & views only!
          Switches: ListRelated=True

Loading Python API...Ready!

Accessing ArcGIS Online/Enterprise...
 - Checking item...
   Relies on Data Item: 4f8efa25bd834ac9ae48ee38058f6bef, 'Test_NWS_Watches_Warnings' (Service Definition)
  Is Relied on by Item: 5144d8a809694e4c82dc8998e6df2857, 'Test_NWS_Watches_Warnings_v1' (Feature Service)
     Is Parent to View: 5144d8a809694e4c82dc8998e6df2857, 'Test_NWS_Watches_Warnings_v1' (Feature Service)

(base) C:\>python OverwriteFS.py esri_livefeeds2 431eab7358ae42909c9b66b29790de95 Test_NWS_Watches_Warnings_C -ListRelat
ed

           Running: OverwriteFS.py, v2.0.0
Python API Profile: esri_livefeeds2
           Item Id: 431eab7358ae42909c9b66b29790de95
        Item Title: 'Test_NWS_Watches_Warnings_C' (for verification)
   Upload Filename: 'None', Touch item & views only!
          Switches: ListRelated=True

Loading Python API...Ready!

Accessing ArcGIS Online/Enterprise...
 - Checking item...
   Relies on Data Item: 52b7d2967e974af6b808e272a2282779, 'Test_NWS_Watches_Warnings_C' (Service Definition)
     Is Parent to View: 5144d8a809694e4c82dc8998e6df2857, 'Test_NWS_Watches_Warnings_v1' (Feature Service)

(base) C:\>
```

- Swapping Layers in a View

In certain instances, you may need to update or Overwrite a Hosted Feature Services that contains a large amount of data. In these cases, a Service Overwrite that takes longer to update the Service than the online Content Delivery Network (CDN) can reasonably support, will cause a service disruption to end users. For this reason, the OverwriteFS script allows for a workflow that can toggle the Layers in a View from one Hosted Feature Service to another, greatly reducing downtime during lengthy updates.

The only requirement for this workflow is that the View must have Multiple-Service relationships added. **See 'Relating Views and Services' topic specific to this requirement**.

Basic concept is that for a View that is related to two Hosted Feature Services, say Service-A and Service-B, the View may point to Feature Service-A while Feature Service-B is idle. During the Overwrite process, Service-B is targeted to receive the Overwrite content while Users continue accessing Service-A (by way of the View). Once Service-B is updated, the View is quickly cleared of all Layers and the newly updated Layers in Service-B are added to the View. Service-A then becomes the idle Service that is targeted for the next overwrite action.


Active Feature Service
Overwrite/Update Target
FS – A
Private Feature Services
FS – B
File Item
File Item
Published from SD, CSV, JSON, fGDB, Excel, ... file
FV – A/B
Published from SD, CSV, JSON, fGDB, Excel, ... file
Feature View(s) Public or Private

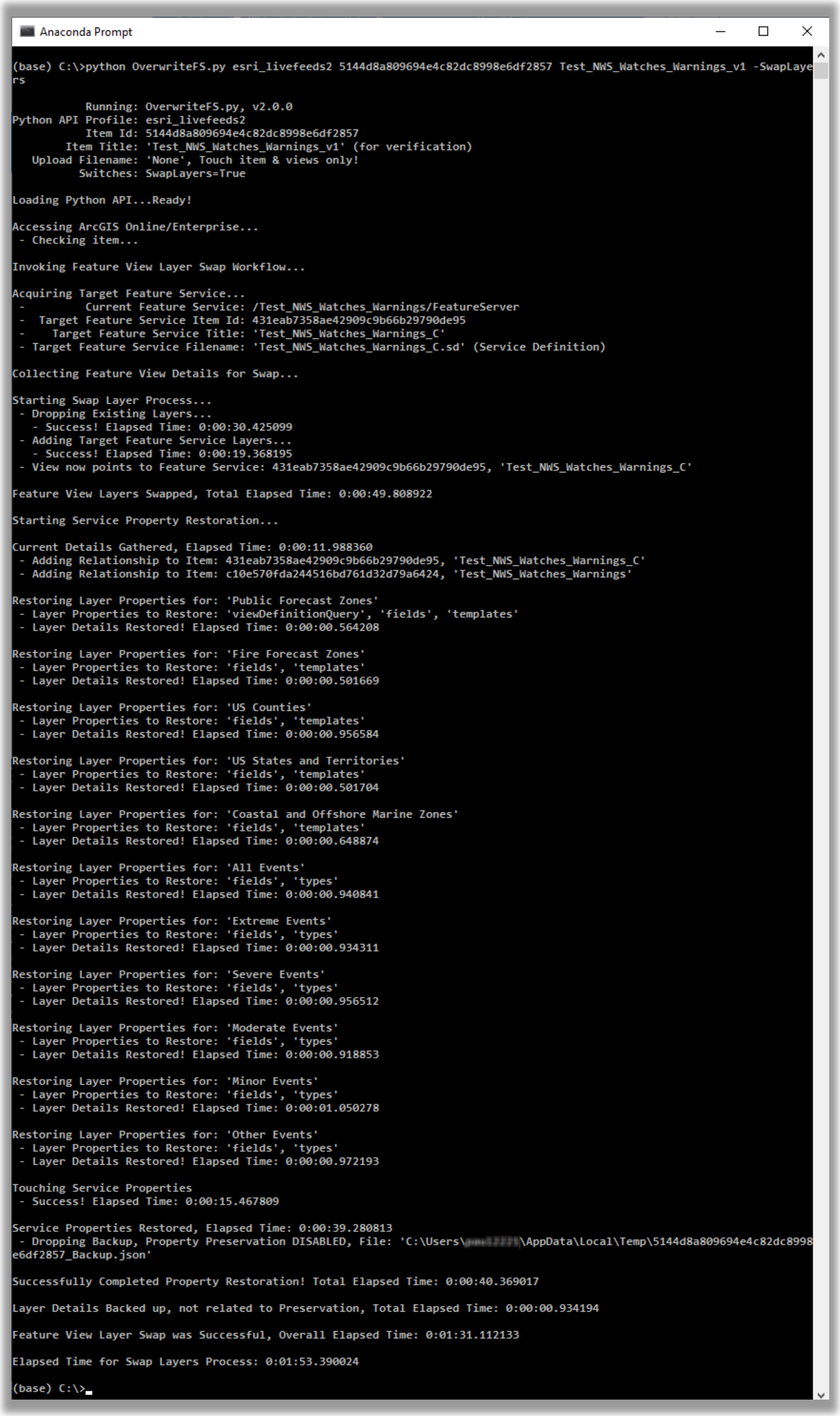
* **Note** * A Hosted Feature View can only point to one Hosted Feature Service at a time!

* **Tip** * Share the View Publicly, but always keep the Services Private. Views are more forgiving and can be directed to different Services should ever you need to replace the Service Item for some reason.

The Swap Layer option can be invoked with or without overwriting the Service. If you need to toggle the Layers of the View so they point to the other Service, simply invoke the OverwriteFS tool and specify the '***-SwapLayers***' Action Switch.

In the following screenshot, we will Swap Layers from Service-A to Service-B on our Test example, which contains 11 Layers in the Service! It also reports the elapsed time for various steps.

This process first identifies the Target Service, then it makes a backup of the Service and/or View details to re-apply layer. Then it Drops the Layers in the View and adds the Layers from the Target Service. Lastly, it re-applies the Layer, Service, and Service Item properties as needed from the backup taken earlier. Remember, Overwriting a Service resets it to the Published state!



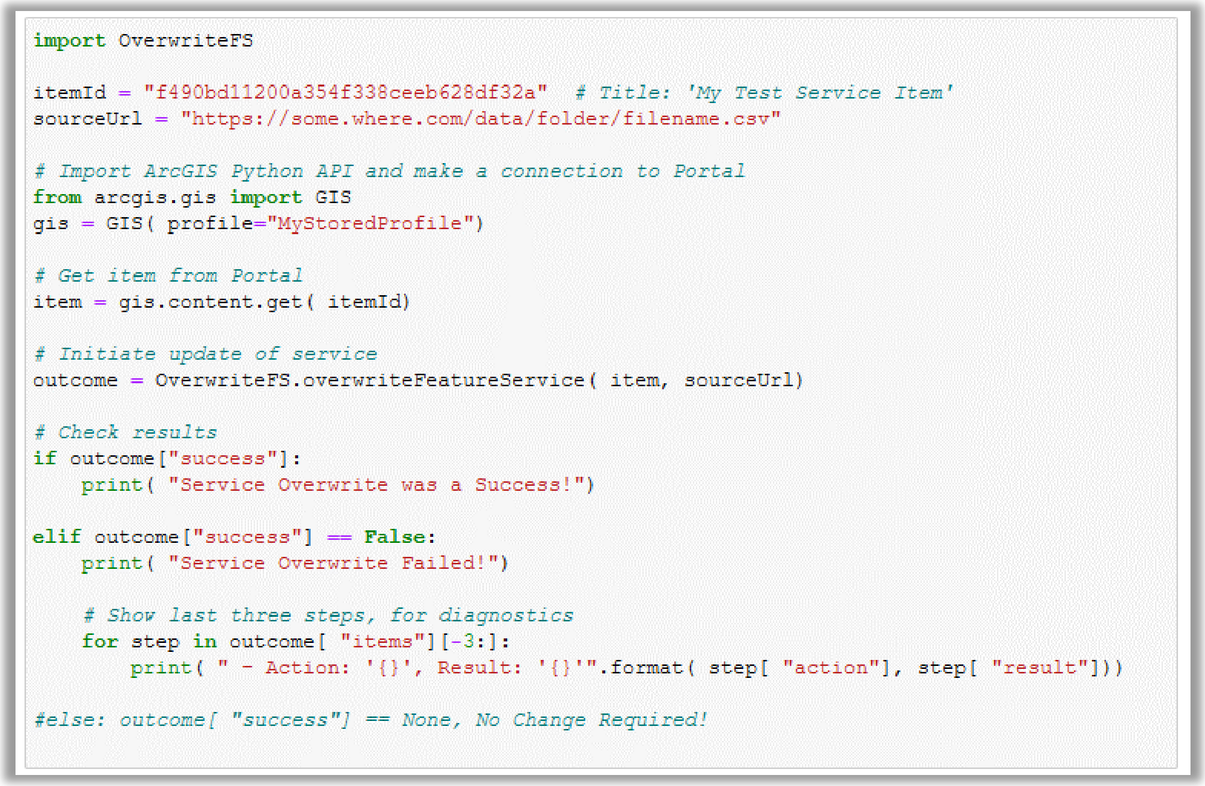
Anaconda Prompt

```
(base) C:\>python OverwriteFS.py esri_livefeeds2 5144d8a809694e4c82dc8998e6df2857 Test_NWS_Watches_Warnings_v1 -SwapLaye
rs

           Running: OverwriteFS.py, v2.0.0
Python API Profile: esri_livefeeds2
           Item Id: 5144d8a809694e4c82dc8998e6df2857
        Item Title: 'Test_NWS_Watches_Warnings_v1' (for verification)
   Upload Filename: 'None', Touch item & views only!
          Switches: SwapLayers=True

Loading Python API...Ready!

Accessing ArcGIS Online/Enterprise...
 - Checking item...

Invoking Feature View Layer Swap Workflow...

Acquiring Target Feature Service...
 -          Current Feature Service: /Test_NWS_Watches_Warnings/FeatureServer
 -  Target Feature Service Item Id: 431eab7358ae42909c9b66b29790de95
 -    Target Feature Service Title: 'Test_NWS_Watches_Warnings_C'
 - Target Feature Service Filename: 'Test_NWS_Watches_Warnings_C.sd' (Service Definition)

Collecting Feature View Details for Swap...

Starting Swap Layer Process...
 - Dropping Existing Layers...
   - Success! Elapsed Time: 0:00:30.425099
 - Adding Target Feature Service Layers...
   - Success! Elapsed Time: 0:00:19.368195
 - View now points to Feature Service: 431eab7358ae42909c9b66b29790de95, 'Test_NWS_Watches_Warnings_C'

Feature View Layers Swapped, Total Elapsed Time: 0:00:49.808922

Starting Service Property Restoration...

Current Details Gathered, Elapsed Time: 0:00:11.988360
 - Adding Relationship to Item: 431eab7358ae42909c9b66b29790de95, 'Test_NWS_Watches_Warnings_C'
 - Adding Relationship to Item: c10e570fda244516bd761d32d79a6424, 'Test_NWS_Watches_Warnings'

Restoring Layer Properties for: 'Public Forecast Zones'
 - Layer Properties to Restore: 'viewDefinitionQuery', 'fields', 'templates'
 - Layer Details Restored! Elapsed Time: 0:00:00.564208

Restoring Layer Properties for: 'Fire Forecast Zones'
 - Layer Properties to Restore: 'fields', 'templates'
 - Layer Details Restored! Elapsed Time: 0:00:00.501669

Restoring Layer Properties for: 'US Counties'
 - Layer Properties to Restore: 'fields', 'templates'
 - Layer Details Restored! Elapsed Time: 0:00:00.956584

Restoring Layer Properties for: 'US States and Territories'
 - Layer Properties to Restore: 'fields', 'templates'
 - Layer Details Restored! Elapsed Time: 0:00:00.501704

Restoring Layer Properties for: 'Coastal and Offshore Marine Zones'
 - Layer Properties to Restore: 'fields', 'templates'
 - Layer Details Restored! Elapsed Time: 0:00:00.648874

Restoring Layer Properties for: 'All Events'
 - Layer Properties to Restore: 'fields', 'types'
 - Layer Details Restored! Elapsed Time: 0:00:00.940841

Restoring Layer Properties for: 'Extreme Events'
 - Layer Properties to Restore: 'fields', 'types'
 - Layer Details Restored! Elapsed Time: 0:00:00.934311

Restoring Layer Properties for: 'Severe Events'
 - Layer Properties to Restore: 'fields', 'types'
 - Layer Details Restored! Elapsed Time: 0:00:00.956512

Restoring Layer Properties for: 'Moderate Events'
 - Layer Properties to Restore: 'fields', 'types'
 - Layer Details Restored! Elapsed Time: 0:00:00.918853

Restoring Layer Properties for: 'Minor Events'
 - Layer Properties to Restore: 'fields', 'types'
 - Layer Details Restored! Elapsed Time: 0:00:01.050278

Restoring Layer Properties for: 'Other Events'
 - Layer Properties to Restore: 'fields', 'types'
 - Layer Details Restored! Elapsed Time: 0:00:00.972193

Touching Service Properties
 - Success! Elapsed Time: 0:00:15.467809

Service Properties Restored, Elapsed Time: 0:00:39.280813
 - Dropping Backup, Property Preservation DISABLED, File: 'C:\Users\[redacted]\AppData\Local\Temp\5144d8a809694e4c82dc8998
e6df2857_Backup.json'

Successfully Completed Property Restoration! Total Elapsed Time: 0:00:40.369017

Layer Details Backed up, not related to Preservation, Total Elapsed Time: 0:00:00.934194

Feature View Layer Swap was Successful, Overall Elapsed Time: 0:01:31.112133

Elapsed Time for Swap Layers Process: 0:01:53.390024

(base) C:\>
```

Python Code Sample

- A standalone example that downloads a file from a Web URL and Overwrites a Service.

'OverwriteFS_Sample.py', included this download, is Python example you can run locally.

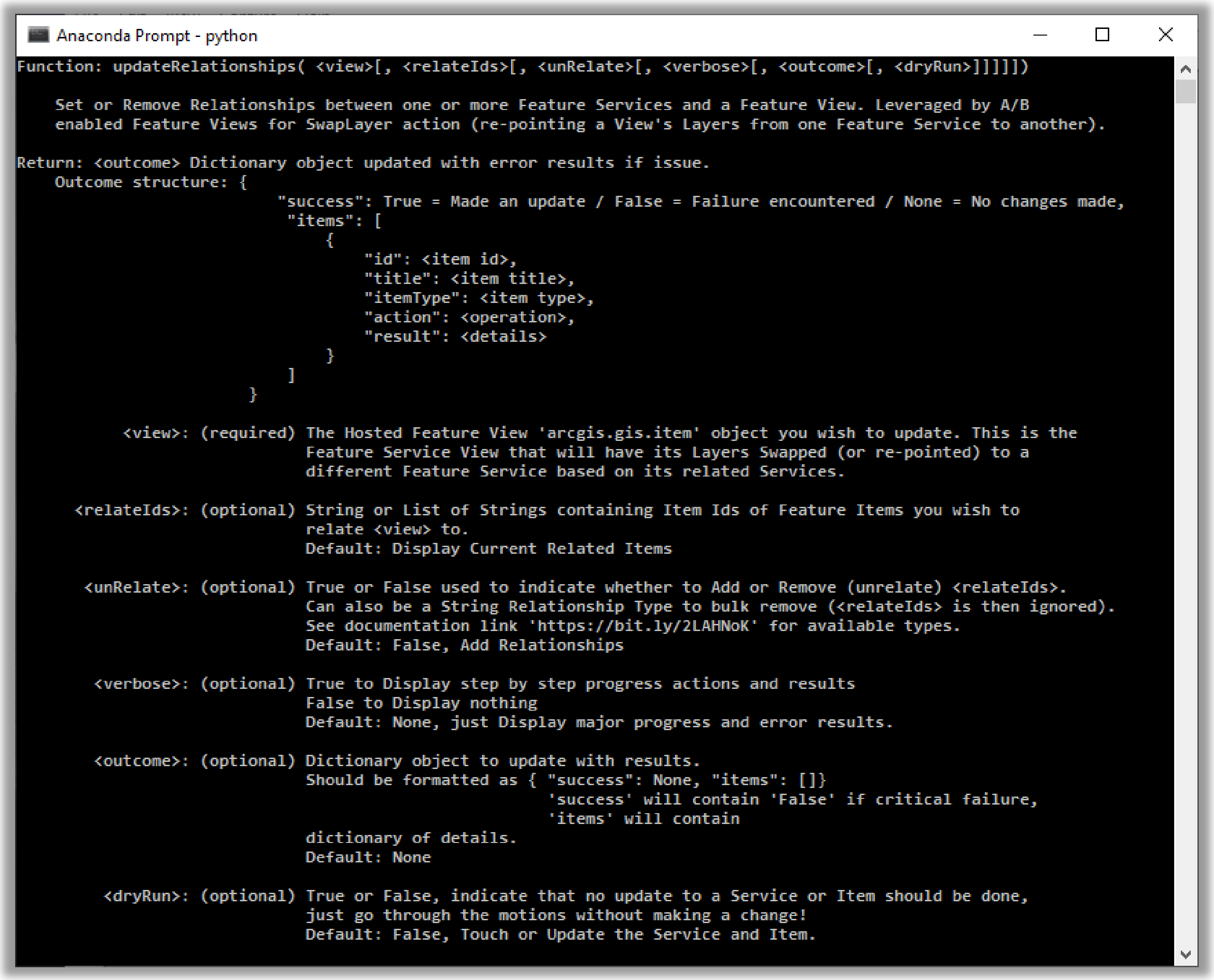


```python
import OverwriteFS

itemId = "f490bd11200a354f338ceeb628df32a"  # Title: 'My Test Service Item'
sourceUrl = "https://some.where.com/data/folder/filename.csv"

# Import ArcGIS Python API and make a connection to Portal
from arcgis.gis import GIS
gis = GIS( profile="MyStoredProfile")

# Get item from Portal
item = gis.content.get( itemId)

# Initiate update of service
outcome = OverwriteFS.overwriteFeatureService( item, sourceUrl)

# Check results
if outcome["success"]:
    print( "Service Overwrite was a Success!")

elif outcome["success"] == False:
    print( "Service Overwrite Failed!")

    # Show last three steps, for diagnostics
    for step in outcome[ "items"][-3:]:
        print( " - Action: '{}', Result: '{}'".format( step[ "action"], step[ "result"]))

#else: outcome[ "success"] == None, No Change Required!
```

Exploring OverwriteFS Functions

- Function: ***updateRelationships***

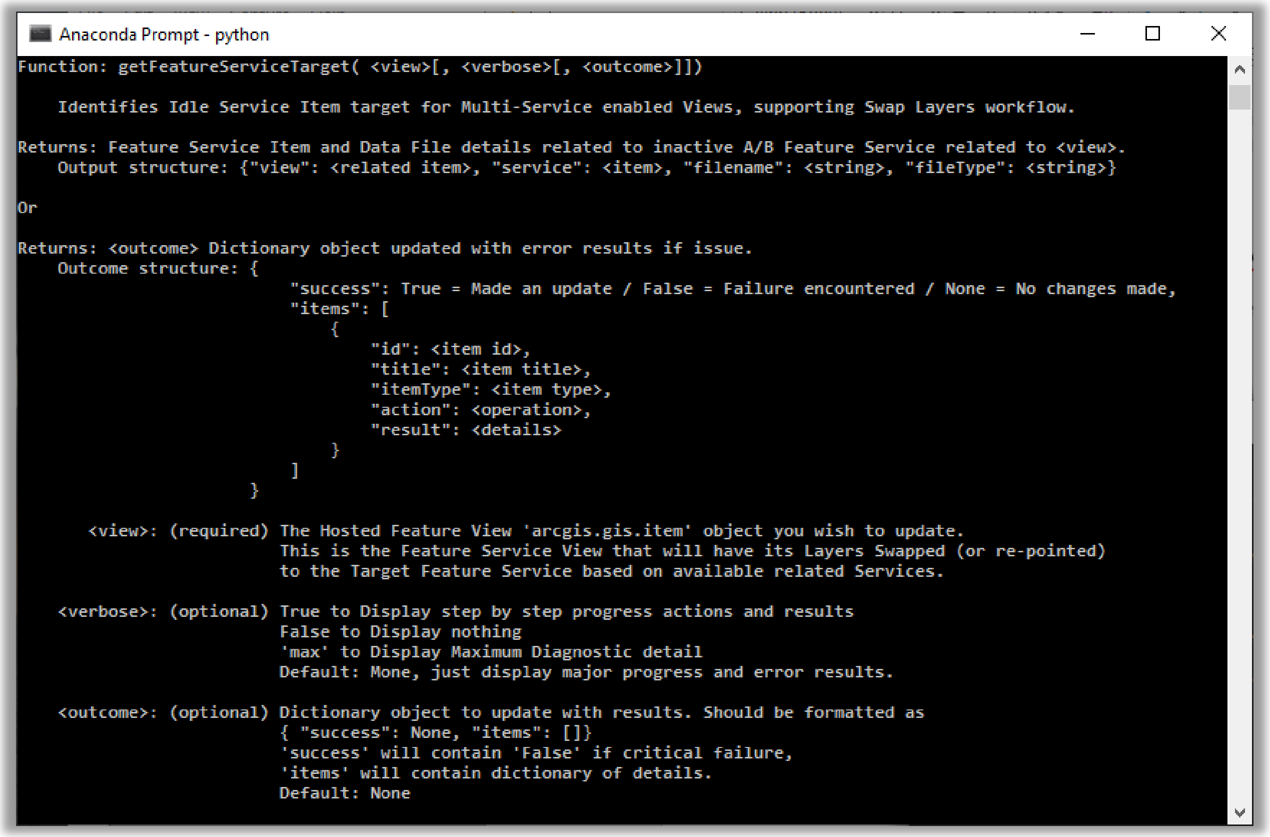
Function that will list, add, or remove a Child to Parent relationships on Items. Typically used for relating Service Items to a View or a Service Item to its File Item.



```
Function: updateRelationships( <view>[, <relateIds>[, <unRelate>[, <verbose>[, <outcome>[, <dryRun>]]]]])

    Set or Remove Relationships between one or more Feature Services and a Feature View. Leveraged by A/B
    enabled Feature Views for SwapLayer action (re-pointing a View's Layers from one Feature Service to another).

Return: <outcome> Dictionary object updated with error results if issue.
    Outcome structure: {
                           "success": True = Made an update / False = Failure encountered / None = No changes made,
                            "items": [
                                {
                                    "id": <item id>,
                                    "title": <item title>,
                                    "itemType": <item type>,
                                    "action": <operation>,
                                    "result": <details>
                                }
                            ]
                        }

           <view>: (required) The Hosted Feature View 'arcgis.gis.item' object you wish to update. This is the
                              Feature Service View that will have its Layers Swapped (or re-pointed) to a
                              different Feature Service based on its related Services.

      <relateIds>: (optional) String or List of Strings containing Item Ids of Feature Items you wish to
                              relate <view> to.
                              Default: Display Current Related Items

       <unRelate>: (optional) True or False used to indicate whether to Add or Remove (unrelate) <relateIds>.
                              Can also be a String Relationship Type to bulk remove (<relateIds> is then ignored).
                              See documentation link 'https://bit.ly/2LAHNoK' for available types.
                              Default: False, Add Relationships

        <verbose>: (optional) True to Display step by step progress actions and results
                              False to Display nothing
                              Default: None, just Display major progress and error results.

        <outcome>: (optional) Dictionary object to update with results.
                              Should be formatted as { "success": None, "items": []}
                                                         'success' will contain 'False' if critical failure,
                                                         'items' will contain
                              dictionary of details.
                              Default: None

         <dryRun>: (optional) True or False, indicate that no update to a Service or Item should be done,
                              just go through the motions without making a change!
                              Default: False, Touch or Update the Service and Item.
```

- Function: ***getFeatureServiceTarget***

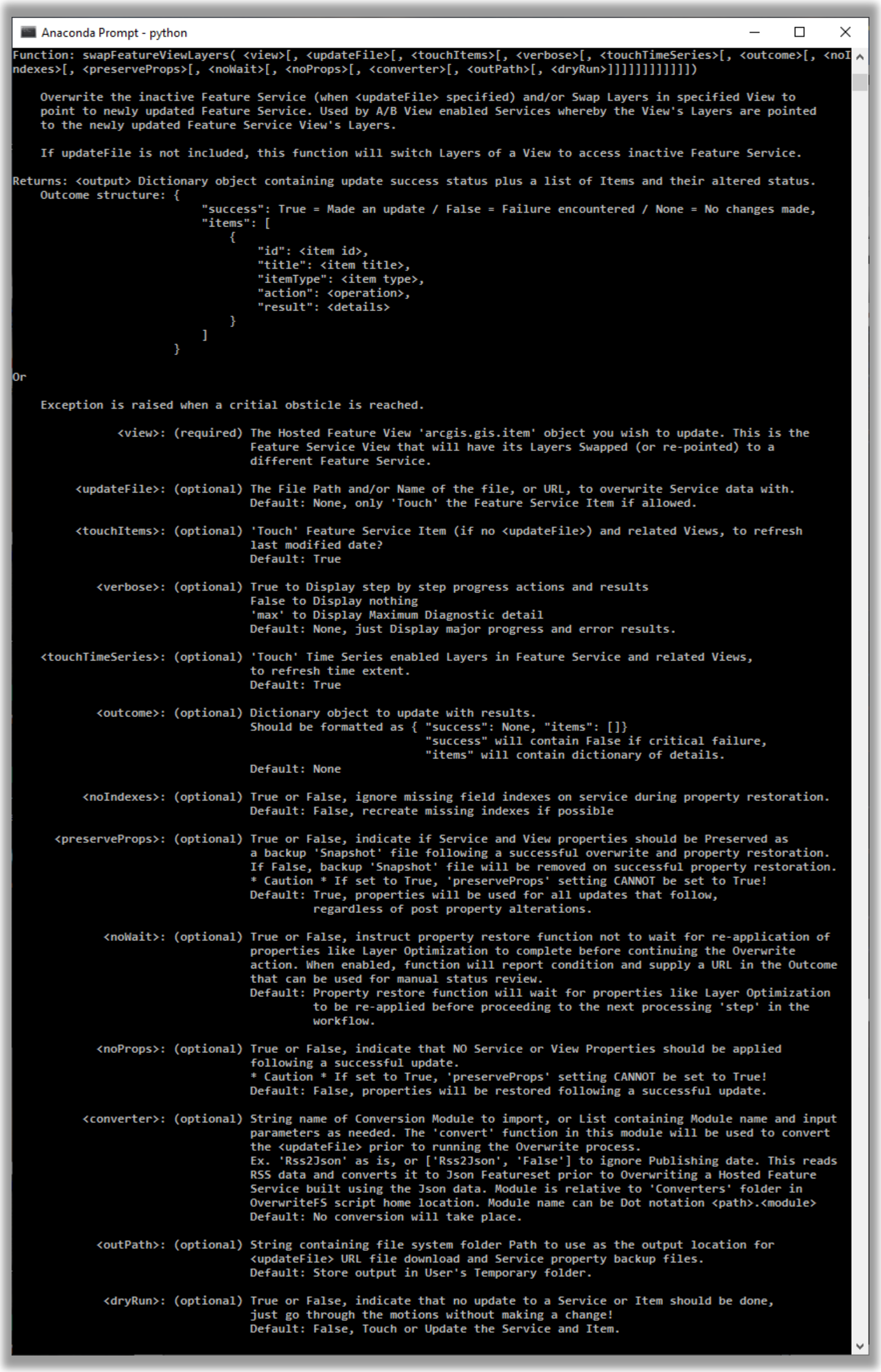
Function that returns the targeted Service Item details when given a View that has Multi-Service relationships.


Anaconda Prompt - python

```
Function: getFeatureServiceTarget( <view>[, <verbose>[, <outcome>]])

    Identifies Idle Service Item target for Multi-Service enabled Views, supporting Swap Layers workflow.

Returns: Feature Service Item and Data File details related to inactive A/B Feature Service related to <view>.
    Output structure: {"view": <related item>, "service": <item>, "filename": <string>, "fileType": <string>}

Or

Returns: <outcome> Dictionary object updated with error results if issue.
    Outcome structure: {
                           "success": True = Made an update / False = Failure encountered / None = No changes made,
                           "items": [
                               {
                                   "id": <item id>,
                                   "title": <item title>,
                                   "itemType": <item type>,
                                   "action": <operation>,
                                   "result": <details>
                               }
                           ]
                       }

       <view>: (required) The Hosted Feature View 'arcgis.gis.item' object you wish to update.
                          This is the Feature Service View that will have its Layers Swapped (or re-pointed)
                          to the Target Feature Service based on available related Services.

    <verbose>: (optional) True to Display step by step progress actions and results
                          False to Display nothing
                          'max' to Display Maximum Diagnostic detail
                          Default: None, just display major progress and error results.

    <outcome>: (optional) Dictionary object to update with results. Should be formatted as
                          { "success": None, "items": []}
                          'success' will contain 'False' if critical failure,
                          'items' will contain dictionary of details.
                          Default: None
```

- Function: ***swapFeatureViewLayers***

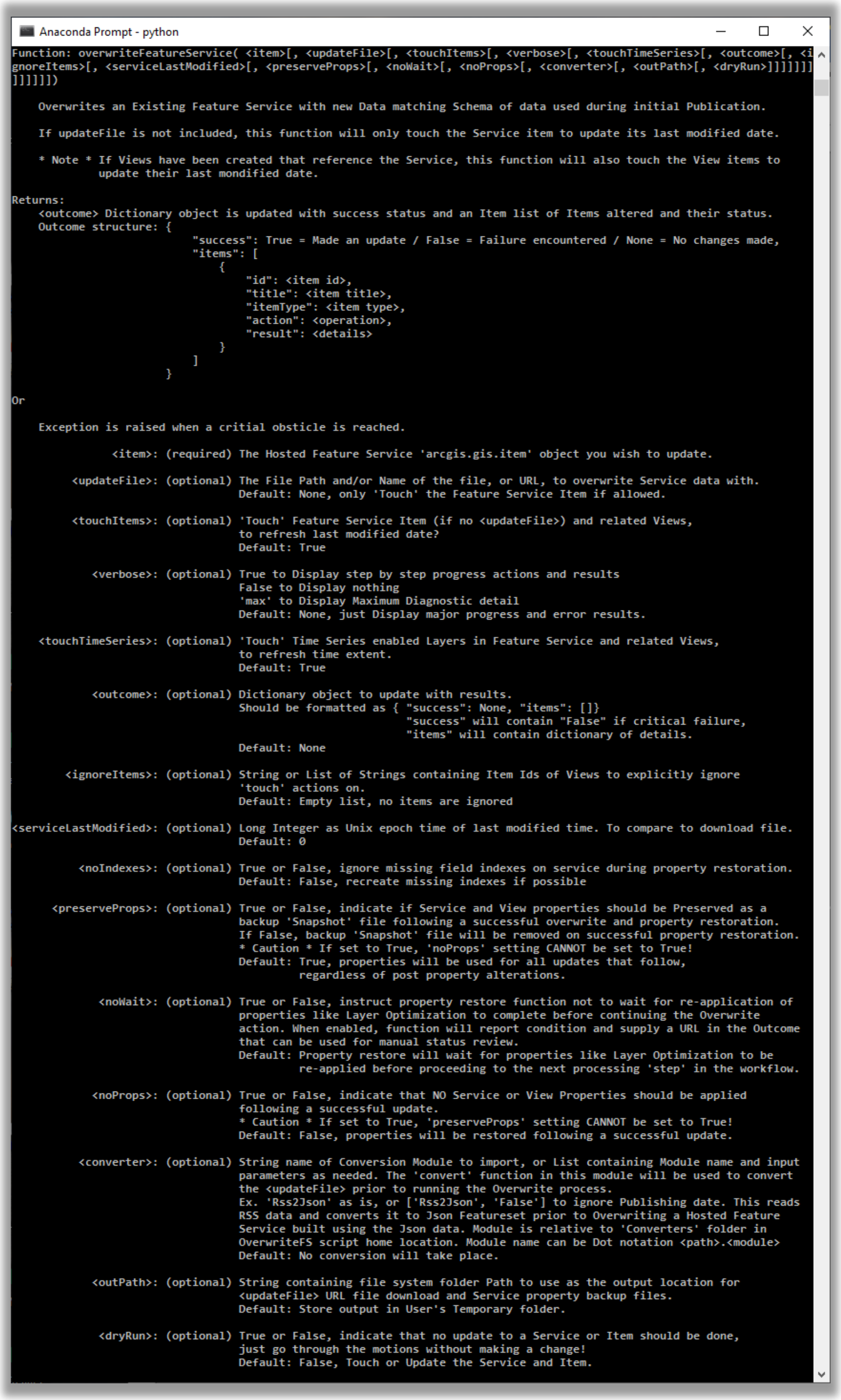
Function that will Swap View Layers to Idle Service Layers or initiate the Overwrite and then Swap Layers depending on the presence of the <updateFile> property.



```
Function: swapFeatureViewLayers( <view>[, <updateFile>[, <touchItems>[, <verbose>[, <touchTimeSeries>[, <outcome>[, <noI
ndexes>[, <preserveProps>[, <noWait>[, <noProps>[, <converter>[, <outPath>[, <dryRun>]]]]]]]]]]]])

    Overwrite the inactive Feature Service (when <updateFile> specified) and/or Swap Layers in specified View to
    point to newly updated Feature Service. Used by A/B View enabled Services whereby the View's Layers are pointed
    to the newly updated Feature Service View's Layers.

    If updateFile is not included, this function will switch Layers of a View to access inactive Feature Service.

Returns: <output> Dictionary object containing update success status plus a list of Items and their altered status.
    Outcome structure: {
                           "success": True = Made an update / False = Failure encountered / None = No changes made,
                           "items": [
                               {
                                   "id": <item id>,
                                   "title": <item title>,
                                   "itemType": <item type>,
                                   "action": <operation>,
                                   "result": <details>
                               }
                           ]
                       }

Or

    Exception is raised when a critial obsticle is reached.

              <view>: (required) The Hosted Feature View 'arcgis.gis.item' object you wish to update. This is the
                                 Feature Service View that will have its Layers Swapped (or re-pointed) to a
                                 different Feature Service.

        <updateFile>: (optional) The File Path and/or Name of the file, or URL, to overwrite Service data with.
                                 Default: None, only 'Touch' the Feature Service Item if allowed.

        <touchItems>: (optional) 'Touch' Feature Service Item (if no <updateFile>) and related Views, to refresh
                                 last modified date?
                                 Default: True

           <verbose>: (optional) True to Display step by step progress actions and results
                                 False to Display nothing
                                 'max' to Display Maximum Diagnostic detail
                                 Default: None, just Display major progress and error results.

   <touchTimeSeries>: (optional) 'Touch' Time Series enabled Layers in Feature Service and related Views,
                                 to refresh time extent.
                                 Default: True

           <outcome>: (optional) Dictionary object to update with results.
                                 Should be formatted as { "success": None, "items": []}
                                                         "success" will contain False if critical failure,
                                                         "items" will contain dictionary of details.
                                 Default: None

         <noIndexes>: (optional) True or False, ignore missing field indexes on service during property restoration.
                                 Default: False, recreate missing indexes if possible

     <preserveProps>: (optional) True or False, indicate if Service and View properties should be Preserved as
                                 a backup 'Snapshot' file following a successful overwrite and property restoration.
                                 If False, backup 'Snapshot' file will be removed on successful property restoration.
                                 * Caution * If set to True, 'preserveProps' setting CANNOT be set to True!
                                 Default: True, properties will be used for all updates that follow,
                                          regardless of post property alterations.

            <noWait>: (optional) True or False, instruct property restore function not to wait for re-application of
                                 properties like Layer Optimization to complete before continuing the Overwrite
                                 action. When enabled, function will report condition and supply a URL in the Outcome
                                 that can be used for manual status review.
                                 Default: Property restore function will wait for properties like Layer Optimization
                                          to be re-applied before proceeding to the next processing 'step' in the
                                          workflow.

           <noProps>: (optional) True or False, indicate that NO Service or View Properties should be applied
                                 following a successful update.
                                 * Caution * If set to True, 'preserveProps' setting CANNOT be set to True!
                                 Default: False, properties will be restored following a successful update.

         <converter>: (optional) String name of Conversion Module to import, or List containing Module name and input
                                 parameters as needed. The 'convert' function in this module will be used to convert
                                 the <updateFile> prior to running the Overwrite process.
                                 Ex. 'Rss2Json' as is, or ['Rss2Json', 'False'] to ignore Publishing date. This reads
                                 RSS data and converts it to Json Featureset prior to Overwriting a Hosted Feature
                                 Service built using the Json data. Module is relative to 'Converters' folder in
                                 OverwriteFS script home location. Module name can be Dot notation <path>.<module>
                                 Default: No conversion will take place.

           <outPath>: (optional) String containing file system folder Path to use as the output location for
                                 <updateFile> URL file download and Service property backup files.
                                 Default: Store output in User's Temporary folder.

            <dryRun>: (optional) True or False, indicate that no update to a Service or Item should be done,
                                 just go through the motions without making a change!
                                 Default: False, Touch or Update the Service and Item.
```

- Function: ***overwriteFeatureService***

Function that will perform the Service Overwrite depending on the presence of the <updateFile> property. Can also initiate the Convert option if present.



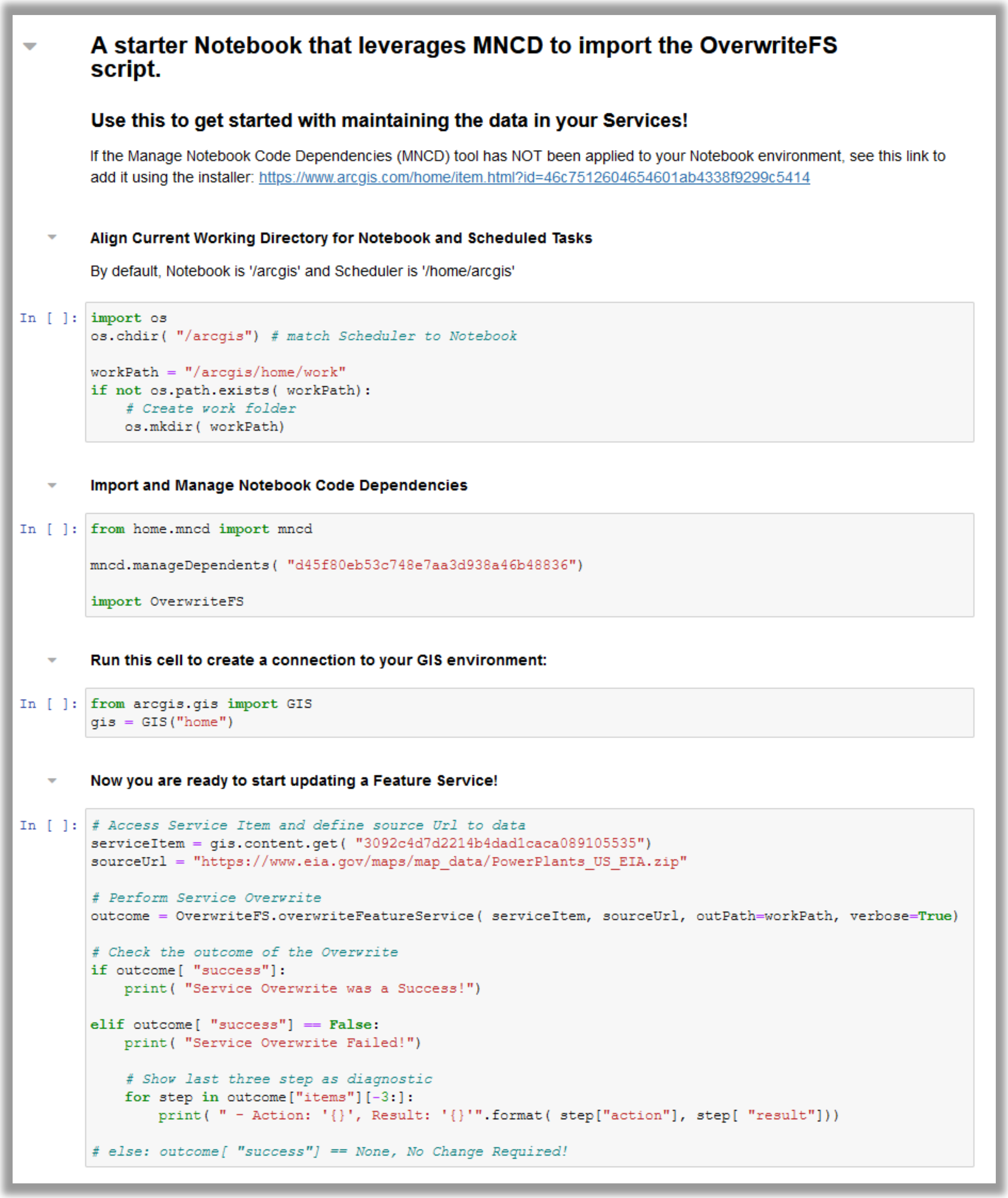
Anaconda Prompt - python

```
Function: overwriteFeatureService( <item>[, <updateFile>[, <touchItems>[, <verbose>[, <touchTimeSeries>[, <outcome>[, <i
gnoreItems>[, <serviceLastModified>[, <preserveProps>[, <noWait>[, <noProps>[, <converter>[, <outPath>[, <dryRun>]]]]]]
]]]]]])

    Overwrites an Existing Feature Service with new Data matching Schema of data used during initial Publication.

    If updateFile is not included, this function will only touch the Service item to update its last modified date.

    * Note * If Views have been created that reference the Service, this function will also touch the View items to
             update their last mondified date.

Returns:
    <outcome> Dictionary object is updated with success status and an Item list of Items altered and their status.
    Outcome structure: {
                           "success": True = Made an update / False = Failure encountered / None = No changes made,
                           "items": [
                               {
                                   "id": <item id>,
                                   "title": <item title>,
                                   "itemType": <item type>,
                                   "action": <operation>,
                                   "result": <details>
                               }
                           ]
                       }

Or

    Exception is raised when a critial obsticle is reached.

               <item>: (required) The Hosted Feature Service 'arcgis.gis.item' object you wish to update.

         <updateFile>: (optional) The File Path and/or Name of the file, or URL, to overwrite Service data with.
                                  Default: None, only 'Touch' the Feature Service Item if allowed.

         <touchItems>: (optional) 'Touch' Feature Service Item (if no <updateFile>) and related Views,
                                  to refresh last modified date?
                                  Default: True

            <verbose>: (optional) True to Display step by step progress actions and results
                                  False to Display nothing
                                  'max' to Display Maximum Diagnostic detail
                                  Default: None, just Display major progress and error results.

    <touchTimeSeries>: (optional) 'Touch' Time Series enabled Layers in Feature Service and related Views,
                                  to refresh time extent.
                                  Default: True

            <outcome>: (optional) Dictionary object to update with results.
                                  Should be formatted as { "success": None, "items": []}
                                                          "success" will contain "False" if critical failure,
                                                          "items" will contain dictionary of details.
                                  Default: None

        <ignoreItems>: (optional) String or List of Strings containing Item Ids of Views to explicitly ignore
                                  'touch' actions on.
                                  Default: Empty list, no items are ignored

<serviceLastModified>: (optional) Long Integer as Unix epoch time of last modified time. To compare to download file.
                                  Default: 0

          <noIndexes>: (optional) True or False, ignore missing field indexes on service during property restoration.
                                  Default: False, recreate missing indexes if possible

      <preserveProps>: (optional) True or False, indicate if Service and View properties should be Preserved as a
                                  backup 'Snapshot' file following a successful overwrite and property restoration.
                                  If False, backup 'Snapshot' file will be removed on successful property restoration.
                                  * Caution * If set to True, 'noProps' setting CANNOT be set to True!
                                  Default: True, properties will be used for all updates that follow,
                                           regardless of post property alterations.

             <noWait>: (optional) True or False, instruct property restore function not to wait for re-application of
                                  properties like Layer Optimization to complete before continuing the Overwrite
                                  action. When enabled, function will report condition and supply a URL in the Outcome
                                  that can be used for manual status review.
                                  Default: Property restore will wait for properties like Layer Optimization to be
                                           re-applied before proceeding to the next processing 'step' in the workflow.

            <noProps>: (optional) True or False, indicate that NO Service or View Properties should be applied
                                  following a successful update.
                                  * Caution * If set to True, 'preserveProps' setting CANNOT be set to True!
                                  Default: False, properties will be restored following a successful update.

          <converter>: (optional) String name of Conversion Module to import, or List containing Module name and input
                                  parameters as needed. The 'convert' function in this module will be used to convert
                                  the <updateFile> prior to running the Overwrite process.
                                  Ex. 'Rss2Json' as is, or ['Rss2Json', 'False'] to ignore Publishing date. This reads
                                  RSS data and converts it to Json Featureset prior to Overwriting a Hosted Feature
                                  Service built using the Json data. Module is relative to 'Converters' folder in
                                  OverwriteFS script home location. Module name can be Dot notation <path>.<module>
                                  Default: No conversion will take place.

            <outPath>: (optional) String containing file system folder Path to use as the output location for
                                  <updateFile> URL file download and Service property backup files.
                                  Default: Store output in User's Temporary folder.

             <dryRun>: (optional) True or False, indicate that no update to a Service or Item should be done,
                                  just go through the motions without making a change!
                                  Default: False, Touch or Update the Service and Item.
```

Online Notebook Code Sample

- A Shared Notebook example that downloads a file from a Web URL and Overwrites a Service.

'Sample OverwriteFS Notebook', https://www.arcgis.com/home/item.html?id=cd5819375aca40a3a7f8b3a269404c2c



A starter Notebook that leverages MNCD to import the OverwriteFS script.

Use this to get started with maintaining the data in your Services!

If the Manage Notebook Code Dependencies (MNCD) tool has NOT been applied to your Notebook environment, see this link to add it using the installer: https://www.arcgis.com/home/item.html?id=46c7512604654601ab4338f9299c5414

Align Current Working Directory for Notebook and Scheduled Tasks

By default, Notebook is '/arcgis' and Scheduler is '/home/arcgis'

```
In [ ]: import os
        os.chdir( "/arcgis") # match Scheduler to Notebook

        workPath = "/arcgis/home/work"
        if not os.path.exists( workPath):
            # Create work folder
            os.mkdir( workPath)
```

Import and Manage Notebook Code Dependencies

```
In [ ]: from home.mncd import mncd

        mncd.manageDependents( "d45f80eb53c748e7aa3d938a46b48836")

        import OverwriteFS
```

Run this cell to create a connection to your GIS environment:

```
In [ ]: from arcgis.gis import GIS
        gis = GIS("home")
```

Now you are ready to start updating a Feature Service!

```
In [ ]: # Access Service Item and define source Url to data
        serviceItem = gis.content.get( "3092c4d7d2214b4dad1caca089105535")
        sourceUrl = "https://www.eia.gov/maps/map_data/PowerPlants_US_EIA.zip"

        # Perform Service Overwrite
        outcome = OverwriteFS.overwriteFeatureService( serviceItem, sourceUrl, outPath=workPath, verbose=True)

        # Check the outcome of the Overwrite
        if outcome[ "success"]:
            print( "Service Overwrite was a Success!")

        elif outcome[ "success"] == False:
            print( "Service Overwrite Failed!")

            # Show last three step as diagnostic
            for step in outcome["items"][-3:]:
                print( " - Action: '{}', Result: '{}'".format( step["action"], step[ "result"]))

        # else: outcome[ "success"] == None, No Change Required!
```

Guidance, Limitations, and Known Issues

- **Guidance**
    - Test, test, test. It is always a good idea to try out new functionality on a Test Service or Item that has no real importance. You should become familiar with script behavior and experiment to see how it interacts with your content and environment. Don't be afraid to break a test Service or Item, you can always discard it!

    - When specifying parameters on the command line, remember that a space is used to separate parameters. If you are entering an Item Title that happens to include a space, be sure to surround the text with double quotes to ensure it is treated as a single parameter! Ex: "My Test Service", and don't confuse a single quote ‘, with a double quote “.

    - When considering schema changes, always 'add' fields to the end of the schema, as not to disrupt existing users. Never 'delete' a field, this will most definitely disrupt users, hide them instead. Some field types can be changed, like a Double replacing an Integer. The Double has a larger capacity and will accept Integer values. An Integer will accept a Short. Always go from small type to a large type and you are not likely to experience an issue.

    - When making schema changes to a View that is supported by Multiple Services, consider updating the Idle Service manually first, without invoking the Swap Layers. This will give you a chance to verify the structural changes for any possible issues, a QA/QC process. You can invoke the Swap Layers action manually as well when the updated Service is vetted! Then repeat the change on the now idle Service. You can always Swap Layers back to the original Service if the View encounters unforeseen issues!

    - When creating multiple Services to use with a View, each Service will require its own File Item with a uniquely named file. Think about naming them with a 'A', 'B' or '1', '2' type of sequence in the title to keep track of which one goes with which Service.

    - When considering creating multiple Views to support a desired content offering, examine the reasoning behind needing multiple Views. Are you needing to hide certain data fields from the target users? Do you need to add a Definition Query to block access to certain rows? Are you wanting to limit access to a given Geography? Is security required to only share the View with certain users? These are all good reasons to leverage a View. But if all you need or want is different Symbology, consider creating a 'Feature Layer' Item instead. This uses the same source Service as a View of Service Item, but it will store the Symbology and Popups in the data property of the Feature Layer Item, separate from the original View or Service. As a bonus, they only create a content Item, no additional Service or duplication of Data. You can create one right from the 'Visualization' tab on any Service or View, whether you own it or not! Plus, these are easily shared and consumed just like a View or Service Item is. If one is created from a Swap Layers supported View, the Feature Layer Item will also benefit from the Swap since it points to the View.

- **Limitations**
  - When Swapping Layers on a View, where the parent Service may have multiple Views, this script does not currently support cascading the Swap to these additional Views. You would need to perform a Swap Layers request on each View. This cascading capability is under consideration for a future release!
  - This script always assumes your Items exist. It is not designed to “create” an Item.
  - Currently, this script does not have the ability to add new layers to a View that did not already have them. Referring to cases where Overwriting a Service may add new Layers because of a schema change. This capability is under consideration for a future release!
  - A Service cannot be Overwritten if it supports a Tile, Vector Tile, OGC, or WFS Service. Damage to these Service type can result.

- **Known Issues**
  - In rare cases, reapplying altered Service or View properties can fail. Not by way of an error, nay they will complete successfully yet never be applied at the back end. We are working to remedy this in future releases, but you can leverage the ‘***-PersistProps***’ option to ensure that critical properties are never lost! Most notable is Post-Publishing Symbology changes. They are applied directly to the Service of a View or Service Item. When Overwriting, the Service details are returned to their original Published state. This forces the script to reapply the most recently backed up details back to the Service, which has the potential of ‘not taking’ on the Service side depending on how busy the environment is at the time.

Release History

- **April 2019, v1.4.1**: Initial public release.
- **April 2020, v1.4.2**: (Internal) Updated to detect if Profile account password exists, to help prevent Task locking up due to 'Enter Password' prompt initiated by Python API if the password is missing!
- **May 2020, v1.4.3**: Finalize check for Profile account password, exit if not populated. Enhanced Profile connection to allow using your active ArcGIS Pro account.
- **July 2020, v1.4.4**: Corrected URL Header access issue during downloads, allowing access to FTP data sources.
- **September 2021, v2.0.0**: Added '***-h***', '***-SwapLayers***', '***-GetTarget***', '***-ListRelated***', '***-AddRelated***', '***-RemoveRelated***', and '***-Convert***' Action Switches. Added '***-OutPath***', '***-NoTimeSeries***', '***-NoIndexes***', '***-NoTouch***', '***-NoWait***', '***-NoProps***', '***-PersistProps***', '***-DryRun***', '***-AllowPWprompt***', '***-LessDetail***', '***-MoreDetail***', and '***-Password***' Option Switches. Added workflow to leverage two Hosted Feature Services supporting one View, alternating Overwrite to Idle Service then Swapping Layers of View to point to newly updated Service. Supports download data Conversion prior to Overwrite. Supports repair of File Item to Service relationship, enabling Service Overwrite capability. Added Service and View property backup and restoration.
- **September 2021, v2.0.1**: Patch to update 'overwriteFeatureService' Function to correctly report outcome of download and conversion when no changes required. Should be reporting a status of None! Also updated command line execution exit value to report -1 on a 'None' status outcome.
- **October 2021, v2.0.2**: Patch to workflow when Service has one or more Views. Overwriting a Service will cause the ArcGIS backend environment to update the View details like Layer extent. We detected an issue with the most recent ArcGIS Online release where following a Service Overwrite the Layer Extent on the Views are set to the first Feature Extent, not the Extent of all Features. This Patch touches the View's Service Details if its Layers are not Time Enabled, correcting the issue. Also included in this release is an update to the 'Converters/Support/datetimeUtils.py' script. See its documentation for full details!