**ArcGIS Online (AGO) Backup Process**

**Overview**

The process was established to ensure that data stored on AGO could be retrieved in the event that staff accidentally deletes content or if a critical error occurs.  The process was designed to scan all of the content on SEMCOG’s account and download/extract the data to a location on SEMCOG’s network. It organizes the content based on the following structure:

**\<user>\<ago\_folder>\<content\_type>\<item\_folder>\<item>**

Each <item> folder also contains a timestamp.txt which is used to determine if the data has changed on AGO. Each time the process is ran it checks the data on AGO and downloads any data that is new, or data that has been modified. It also deletes from our archive any data that has been deleted.

**Locations**

**Process:** Y:\Projects\GIS\Data\ago\_backup\ago\_backup.py

**AGO Login for Process:** Y:\Projects\GIS\Data\ago\_backup\passwords.py

**Data:** Y:\Projects\GIS\Data\ago\_backup\Content\

**Schedule**

The process runs on DEVRDP as a Scheduled Task (named “Back up ArcGIS Online Content”) at 4:00am everyday.

The task has one action which runs the following **program**:

"C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe"

Note: the script has to use this Python location so that it has access to the [ArcGIS API for Python](https://esri.github.io/arcgis-python-api/apidoc/html/)

With the following **argument**:

\\semcog-fs\DA\Projects\GIS\Data\ago\_backup\ago\_backup.py

And **starts** in the following location:

\\semcog-fs\DA\Projects\GIS\Data\ago\_backup\

And executed with the “misiuk” user Windows account.

**git**

The process uses [git](https://git-scm.com/) to keep track of historical changes and “commits” to a git repository every time it is ran. Any content that is backed up using this process will be saved in the git repository. This means that even if a file is deleted or has incorrect edits made can be restored by looking back in the git history.

**General Notes:**

**Feature Services**

In order to ensure that we backup feature services correctly they need to be handled differently from everything else. If we use the typical “download” method, the source data of the feature service is not downloaded, only a text file with some configuration information if applicable. To get past this, we use the “export” method so that the feature service is backed up properly. The exported file geodatabase is saved in the same directory as the feature service text file.

The date modified of the feature service is not changed if data is edited or changed at all. So we need to parse the properties of the layers of the service to check the lastEditDate. The lastEditDate is then used in the timestamp file and allows us to ensure that we archive changes made to feature services.

**Code Attachments**

By default, some content like Web App Builder apps, will have empty Code Attachments alongside them. The download method fails when it gets to this item. In order for the script to run without failing, we need to ignore Code Attachments.

**Feature Services in table form**

If we host a tabular dataset as a feature service (e.g. Park Attributes) we cannot extract it to a File Geodatabase. This is a special case that requires us to export to a csv.