

Cornwall Park Trust Board 

Related Fields Management Tool

16/12/2016

Fraser Hand

V1.0

Table of Contents

[1 Related Fields Management Tool 3](#_Toc470013759)

[1.1 Background 3](#_Toc470013760)

[1.2 Summary 3](#_Toc470013761)

[1.3 Prerequisites 4](#_Toc470013762)

[1.4 Points to Note around Prerequisites 4](#_Toc470013763)

[1.5 Script Installation / Deployment 6](#_Toc470013764)

[1.6 Additional Note 7](#_Toc470013765)

[1.7 Application Architecture 7](#_Toc470013766)

[1.8 Process Flow 9](#_Toc470013767)

[1.9 Configuration 10](#_Toc470013768)

[1.10 Running the Application 12](#_Toc470013769)

[1.11 Making Changes to the Application 13](#_Toc470013770)

[1.11.1 Changes to the Client 13](#_Toc470013771)

[1.11.2 Changes to the Python Toolbox 13](#_Toc470013772)

[2 Python Object Model 14](#_Toc470013773)

Change Log

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Change Profile |
| 1.0 | 16/12/2016 | Fraser Hand | Initial Release |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Distribution List

|  |  |  |
| --- | --- | --- |
| Name | Organisation | Role |
| Marc Graham | Eagle | Technical Lead |
| Nick Duke | Eagle | Account Manager |
| Harvey Wong | Eagle | Project Manager |
|  |  |  |
|  |  |  |
|  |  |  |

1. Related Fields Management Tool
   1. Background

The Cornwall Park Trust Board (CPTB) has implemented an Esri ArcGIS Online portal to enable the management of various spatial datasets found within the park. Due to the size of the organisation and the current organisational requirements ArcGIS Online has been selected as the delivery platform for these datasets. This takes advantage of Software as a Service benefits such as not having to manage any on premise infrastructure or application stack components such as geodatabases.

* 1. Summary

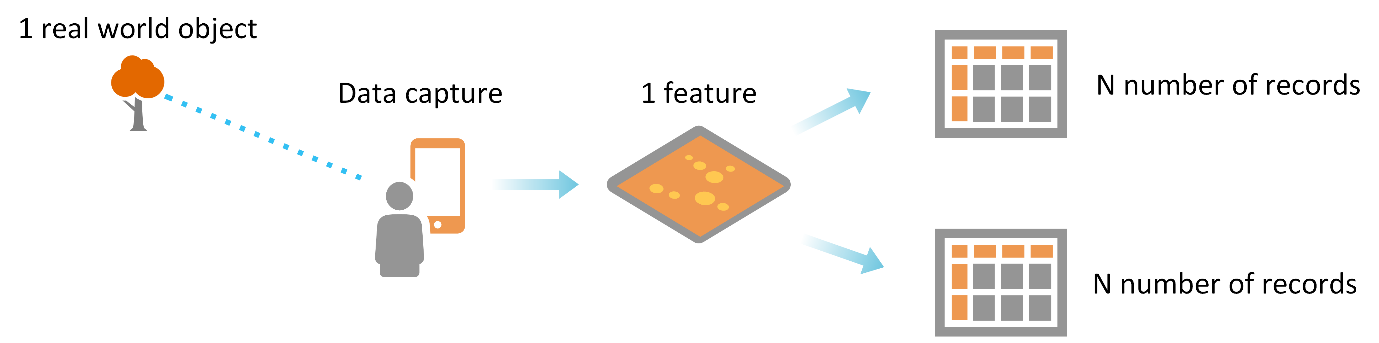
CPTB have uploaded their spatial datasets to ArcGIS Online and have exposed these as hosted feature services with full editing capability. This allows the data to be visualised as Web Maps within the ArcGIS Online environment through the default Map Viewer, created and shared Web Apps or any of the integrated Esri apps (including mobile apps).

CPTB maintain park data (such as trees) through Collector for ArcGIS and this extends to creating Maintenance and Condition reports. This data is then used in a business intelligence methodology to show items such as asset depreciation, assets requiring maintenance with associated cost and the current condition of assets.

Due to the way ArcGIS Online handles relationships between spatial (trees) and non-spatial (reports) data, not all current data will be easily visualised through the Esri clients. Each spatial feature (1) can have multiple (n) related condition or maintenance reports and the latest data is not easily accessible.

Therefore in order to make the latest data easily accessible and visible to the mobility users, related data will be copied back to the base feature class and stored in particular fields so users can see the latest value for a range of data (Condition Report for example).

This workflow is defined below:



As new condition and maintenance reports are created, the data displayed in the parent feature may be out of date. In order to maintain this a nightly process will run across the hosted feature services and update each feature with the latest related data.

This application has been scripted in Python and will be required to be run from a client machine which has internet access. By choosing a pure Python approach the following benefits are realised:

* Python is an open scripting language that is easy to install, use, extend and maintain.
* The report can be run from any machine that has Python installed. ArcGIS and the arcpy installation are not required.
* Python has the benefit of being a scripting language and therefore it is easy to update applications as they are all simple text files. This can be achieved in any text editor. This means the framework can be easily updated.
* ArcGIS Desktop installs Python as part of its installation. If the script runs from a machine with ArcGIS Desktop install Python is already available.
  1. Prerequisites

In order to deploy the tool the prerequisites are:

* Python 2.7.5+ (you can get the version from the command line)

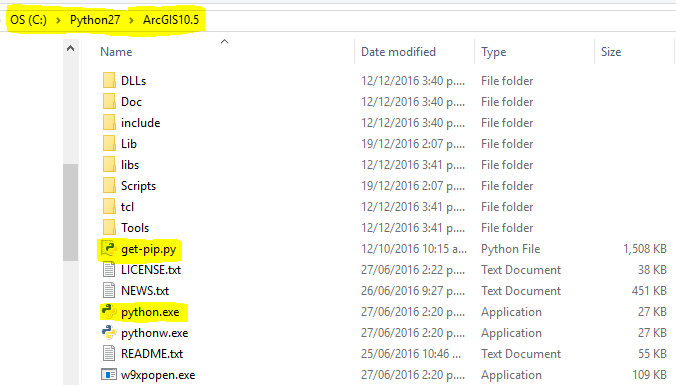


* ArcREST 3.5.4 (<http://esri.github.io/ArcREST/> https://github.com/Esri/ArcREST)
* Xlsxwriter (http://xlsxwriter.readthedocs.io/)
* Pip Python package installer (to install above packages)
  1. Points to Note around Prerequisites

The directory containing the scripts is only one piece of the tool installation. The python script library uses the ArcREST and Xlsxwriter library, and as these are not standard Python libraries and thus need to be installed in the hosting machines Python modules location (site packages). Python has a package manager to make this type of thing simple called pip (<https://pypi.python.org/pypi/pip>). If pip isn’t installed by default with Python (vanilla or from ArcGIS installs) it will need to be installed into the machines Python installation. This is detailed here <https://packaging.python.org/installing/> and it involves downloading and running a python file (get-pip.py). Pip installation can be checked by looking in the <Python install dir>\Scripts and seeing if pip.exe etc. exists.

To install pip get the install from here <https://bootstrap.pypa.io/get-pip.py>, and to make the process easy place it in the directory with python.exe. Once you have done this run the following command:

python get-pip.py

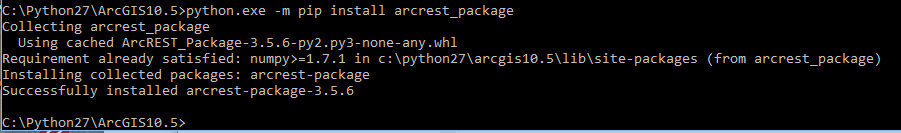




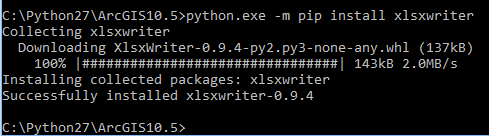
This will go through the process of installing pip and setup tools for the Python installation. Once pip has been installed you can then install the other modules.

Note pip is in the Scripts directory so you will need to tell Python to run it as a module.

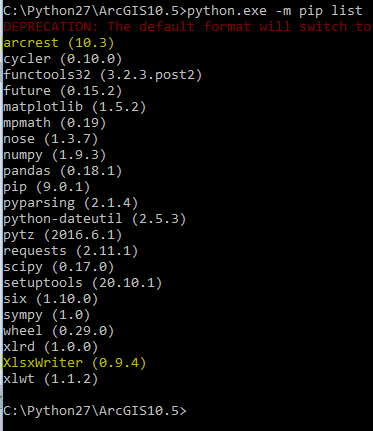
python.exe –m pip install arcrest\_package



Do the same for the Xlsxwriter module



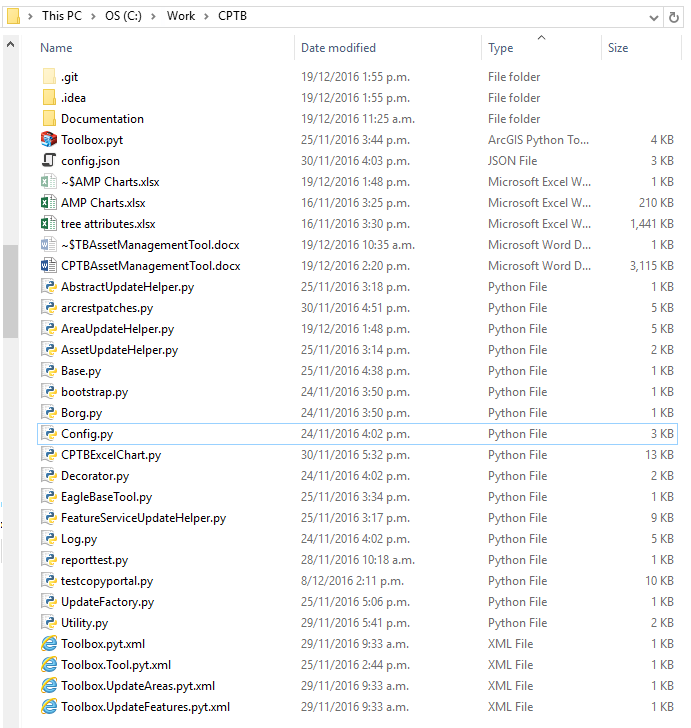
Once the install is finished you can check with pip list



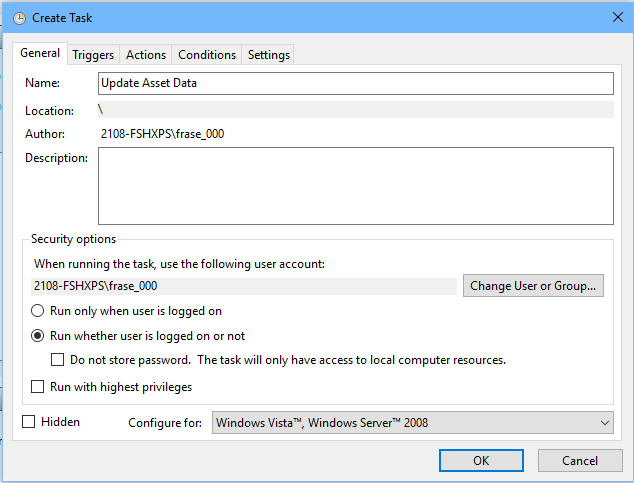
Once this is installed then the script is ready to run.

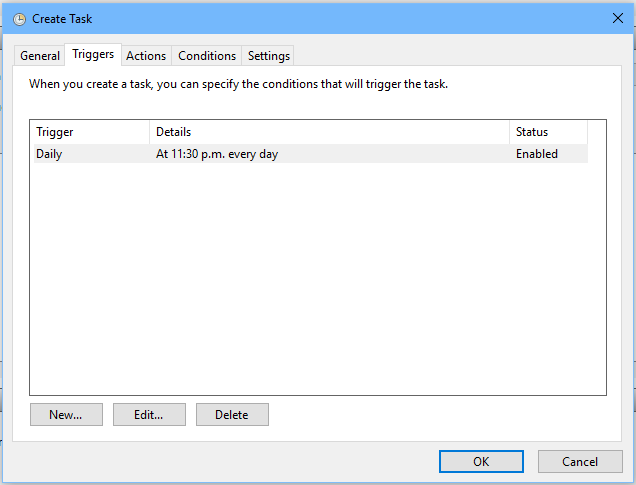
* 1. Script Installation / Deployment

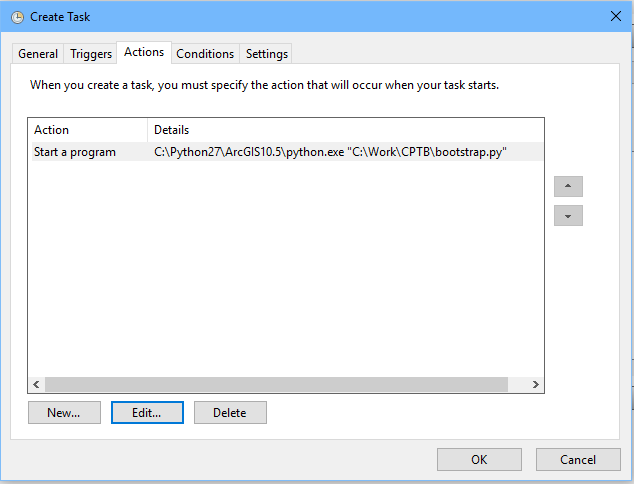
The scripts can be deployed to a well-known location and can be scripted to be run from a scheduled task. In the example below the script has been extracted to C:\Work\CPTB and the main file to execute is c:\Work\CPTB\bootstrap.py. There is no installers or compilation needed.



To automate the process the scheduled task would be set up as below







* 1. Additional Note

Since the application interacts directly with the ArcGIS Online sharing and portaladmin APIs it is assumed that the machine that runs / hosts the application will have internet access.

* 1. Application Architecture

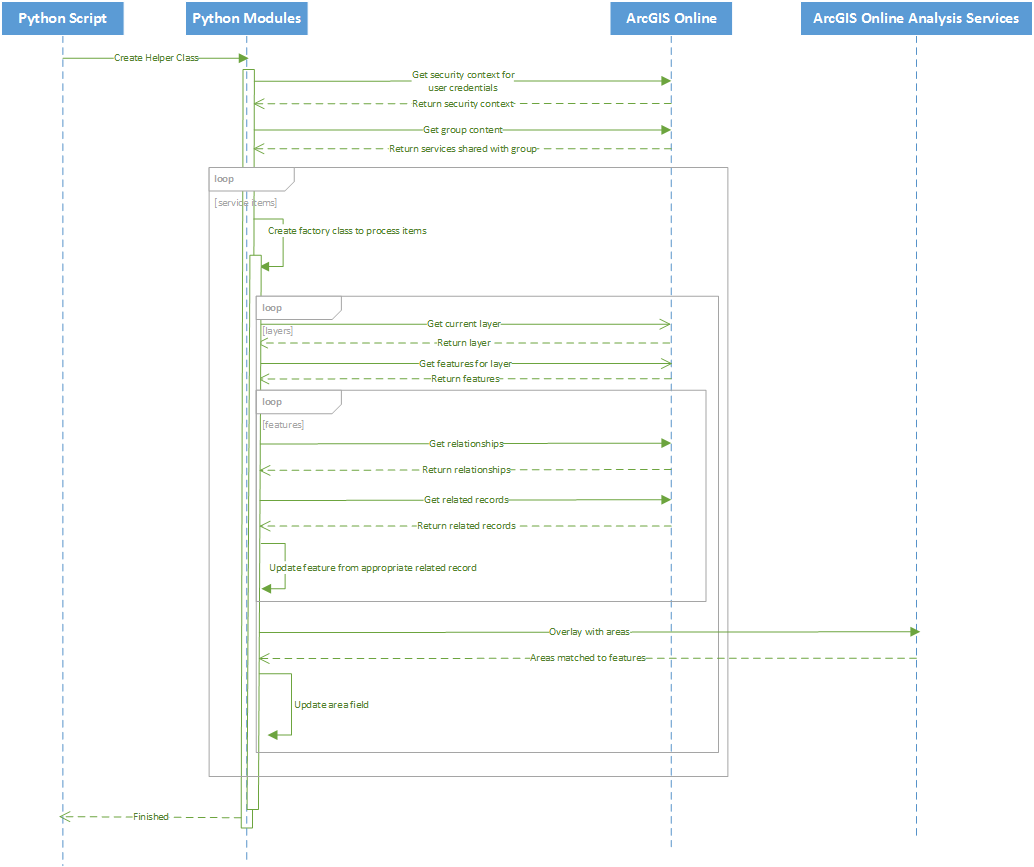
The current application consists of

* Python classes
* Configuration file
* Python Toolbox (this can be used inside of ArcGIS Desktop to call each of the functions individually)

The application contains the following script files:

|  |  |
| --- | --- |
| **Script Name** | **Description** |
| AbstractUpdateHelper.py | Base class for creating various helpers with a defined structure. |
| arcrestpatches.py | Patches for the Esri ArcREST library – subclasses from the base Esri classes with fixes (so as not to affect the core Esri library). |
| AreaUpdateHelper.py | Class for updating the AREA field on a given layer. |
| AssetUpdateHelper.py | Wrapper class for executing the update of given fields for a given layer. |
| Base.py | Parent class with common functionality for child classes. |
| bootstrap.py | Entry point into the application – should be the file that is executed manually or from a scheduled task. |
| Borg.py | Singleton class for shared application configuration |
| config.json | Application configuration |
| Config.py | Application configuration handler – access to logger |
| CPTBExcelChart.py | Creates a set of XSLX files with defined charts and uploads to ArcGIS Online |
| Decorator.py | Provides functions that can be decorated on other functions so that they are  called before the decorated function |
| EagleBaseTool.py | Provides common functionality for child tools within a toolbox |
| FeatureServiceUpdateHelper.py | Updates features in a feature service based on related records for a given set of fields |
| Log.py | Logger class – handles various log levels to console or file |
| Reporttest.py | Test file for instantiating the XLSX functionality |
| Toolbox.pyt | Toolbox which exposes the functions individually so they can be executed from ArcGIS Desktop |
| UpdateFactory.py | Factory pattern for creating updaters |
| Utility.py | Various helper functions |

The following sequence diagram details the series of events and interactions between the majority of the Python classes and the bootstrap Python script.



* 1. Process Flow

The main application executes the following workflow:

* Bootstrap is initiated and creates an AssetUpdateHelper object and executes the update process on it.
* AssetUpdateHelper gets all group items from ArcGIS Online (Automation group) and for each calls update features on the factory that the particular item is passed to.
* UpdateFactory creates the appropriate updater (in this case Feature Service Updater)
* FeatureServiceUpdateHelper creates security context for ArcGIS Online based on configured parameters (username / password). Caches the Area service and analysis overlay task.
* Feature Update Service updates features in a two stage process:
  + Loops over all feature layers and tables present in the service and updates records based on related records field values and configured logic
  + Updates all features with the current area they reside in based on using the ArcGIS Online analysis tools. Note this does involve the use of service credits on ArcGIS Online.
* The process for updating features / records based on related records is as follows:
* For the particular layer all relationships are requested. Then for each feature every related record is requested. Once these have been found the values for each related field is found and a result is selected based on configured criteria. This may be the latest, the oldest which hasn’t been completed etc. Once these values have been found they are updated on the feature / record. Every relationship and every feature / record is processed.
* Once this process is finished, the layer is passed to the ArcGIS Online Overlay task (part of the Analysis tools) and overlaid with the Areas feature service. This returns groups of features by Area. The area name is then updated on each feature.

Due to the nature of every feature / record being processed and checking multiple related records, before performing overlays, the process can take some time to execute. Therefore it is recommended that the process is run outside of business hours to ensure data isn’t being updated while the process is executing.

* 1. Configuration

The application is driven from one main configuration file. This contains dynamic parametrisation which is loaded at runtime. The parameters are described below

{  
 **"logging"**: { *This section defines how the application will log*

**"logtofile"**: **"true"**,*true logs to file false logs to memory*  
 **"loglevel"**: **"debug"** *level of logging from info through to debug*},  
 **"agolconfig"**: { *this section describe how to connect to ArcGIS Online*  
 **"org\_url"**: **"https://cptb.maps.arcgis.com"**, *ArcGIS Online orgainsational URL*  
 **"username"**: **"m.llewellyn"**, *ArcGIS Online organisational user name*  
 **"password"**: **"cptb1234"**, *ArcGIS Online organisational password*  
  
 **"security\_type"**: **"Portal"***This is set as part of ArcREST api – leave as Portal*},  
 **"reports"**: { *this section is for the Excel replort generation*  
 **"reportfolder"**:**"Replorts"**, *Folder in ArcGIS Online organisation (as part of that user) to store the excel replorts*  
 **"sortfield"**: **"Latest\_Condition\_Grade"**,  
 **"skipfields"**: [ *skip fields from process returned results from ArcGIS Online*  
 **"OBJECTID"**,  
 **"GLOBALID"** ],  
 **"reportset"**: { *lookups for creating the replort charts and graphs*  
 **"conditiongrades"**: {  
 **"one"**: [  
 **"Grade"**,  
 **"Latest\_Condition\_Grade"** ],  
 **"title"**:**"Condition Grades"** },  
 **"assettype"**:{  
 **"one"**:[**""**,**"Type"**],  
 **"title"**:**"By Asset Type"** },  
 **"materialtype"**:{  
 **"one"**:[**""**,**"Material"**],  
 **"title"**:**"By Material Type"** },  
  
 **"totalrenewals"**: {  
 **"one"**:  
 **"Estimated\_Renewal\_Date"**,  
 **"two"**:  
 **"Replacement\_Cost"**,  
 **"yaxis"**: **"Cost"**,  
 **"title"**:**"Total Renewals ($)"** }  
 }  
 },  
 **"esrifinishcodes"**: [ *esri codes to check when a job is finished (analysis task)*  
 **"esriJobSucceeded"**, *these are predefined so do not change*  
 **"esriJobFailed"**,  
 **"esriJobTimedOut"**,  
 **"esriJobCancelled"**,  
 **"esriJobDeleted"** ],  
 **"assetgroup"**: **"90a105ec72ab4d3288d501a5aa712d50"**,*group id in ArcGIS Online*  
 **"assetgroupname"**: **"Asset Automation"**,*group name in ArcGIS Online*  
 **"areaupdate"**: {  
 **"areaservice"**: **"https://services6.arcgis.com/sNkVulJvjhuLzi0I/ArcGIS/rest/services/Area\_Name\_Simple/FeatureServer/0"**, *this is the feature layer with the park areas*  
 **"queryfield"**: **"AreaName"**, *field to return on the above service*  
 **"updatefield"**: **"Area"** *field to update on the target layers – must be consistent across all feature services / layers*},  
 **"related\_fields"**: { *this is the lookups for the related fields. For each lookup type we have the parent feature class field to update and the related table field to look for*  
 **"maintenance"**: {  
 **"fieldlookups"**: {  
 **"Scheduled\_Maintenance\_Date"**: **"Next\_Maintenance\_Date"**,  
 **"Scheduled\_Maintenance\_Descripti"**: **"Next\_Maintenance\_Description"** },  
 **"where"**: [ *conditional query to apply*  
 **"Completed\_Maintenance\_Date"**,  
 **null** ],  
 **"select"**: **"oldest"**,  
 **"sortby"**: **"Scheduled\_Maintenance\_Date"** },  
 **"condition"**: {  
 **"sortby"**: **"Condition\_Report\_Date"**,  
 **"select"**: **"newest"**,  
 **"fieldlookups"**: {  
 **"Condition\_Report\_Date"**: **"Latest\_Condition\_Date"**,  
 **"Condition\_Report\_Grade"**: **"Latest\_Condition\_Grade"**,  
 **"Condition\_Report\_Description"**: **"Latest\_Condition\_Description"**,  
 **"Annual\_Risk\_Of\_Harm"**: **"annual\_risk\_of\_harm"**,  
 **"Probability\_Of\_Failure"**: **"probability\_of\_failure"** }  
 }  
 },  
 **"valuation\_field\_lookups"**: {  
 **"installdate"**: **"Installation\_Date"**,  
 **"installcost"**: **"Installation\_Cost"**,  
 **"renewaldate"**: **"Estimated\_Renewal\_Date"**,  
 **"currentvalue"**: **"Current\_Value"** }  
}

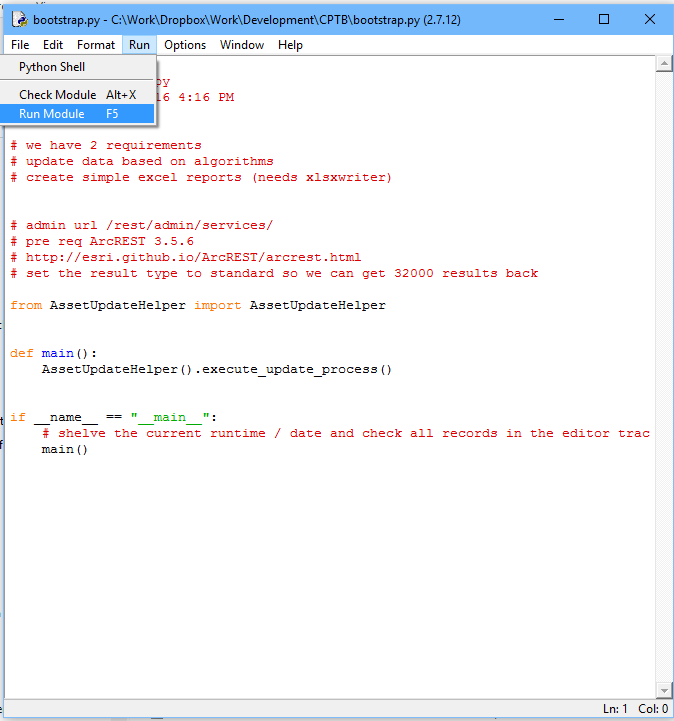
The configuration can be extended at any time and the lookup fields can be updated as required. However it is recommended that before making any changes the full impact of changing the configuration will have on the script operation.

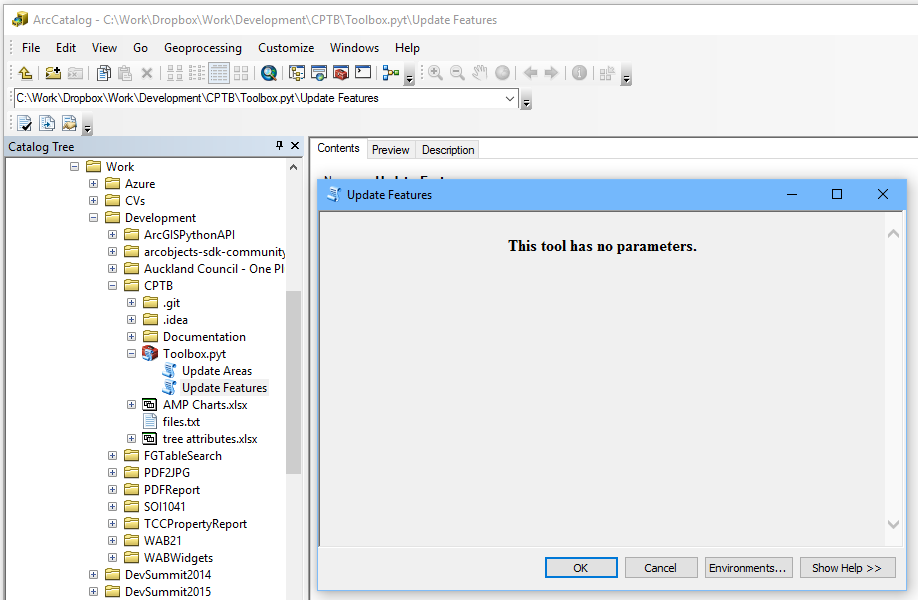
* 1. Running the Application

The python script just needs to be copied to a folder on disk somewhere. It doesn’t required any installation and there is no build process. The script just needs to be executed. This can happen in a number of ways:

* From the command line (executing bootstrap.py)
* From a scheduled task (see above)
* From a Python UI (e.g. IDLE) where the script is executed (opening bootstap.py and running the module)
* From the ArcGIS Desktop toolbox wrappers







* 1. Making Changes to the Application

The application is made up of 3 logical components

* The bootstrap.py script which acts as the client
* Shared Python class / module scripts
* The python toolbox which can be used from ArcGIS Desktop
  + 1. Changes to the Client

Changes can be made directly to the Python on the host machine (as well as the config.json) and these will be picked up on the next run of the application. Delete any .pyc files that may exist before re-running the client application (bootstrap.py).

* + 1. Changes to the Python Toolbox

Changes can be made directly to the Python on the host machine and the toolbox just has to be refreshed in ArcGIS Desktop.

1. Python Object Model

