TransferAttachements.py

This script tool will transfer attachments from features in one feature class to corresponding features in another feature class. This can be handy if you need to change the schema or geometry type of an existing feature class or if you need to merge two feature classes. It is an alternative to the “download each attachment and re-upload to the new feature by hand” method that is rather tedious and boring.

Attachments use the object ID as the join field between the feature class and the attachment table. This tool uses a match field (that must be created and populated prior to running the tool) to create an identifier for each feature that is used to translate the object ID of the source feature class to the object ID of the destination feature class. The attachments are then automatically downloaded to the specified scratch folder and then added to the destination’s attachment table. We cannot use the object IDs directly because they can change value when some geoprocessing tools are run—we can’t guarantee they will remain the same, and they don’t copy over if you copy features from one feature class to the next.

Source__Attach

Att_name	*Rel_oid*
Photo1.jpg	15

⇔

Source FC

OID	*MatchID*
15	200

⇨

Dest FC

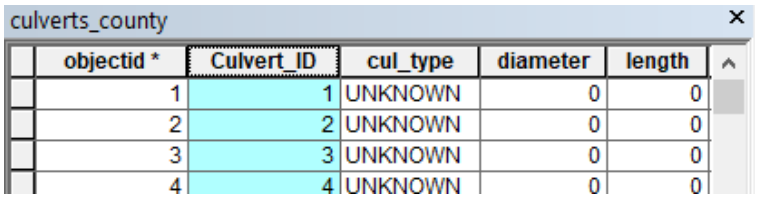
MatchID	*OID*
200	67

⇔

Dest__Attach

Rel_oid	*Att_name*
67	Photo.jpg

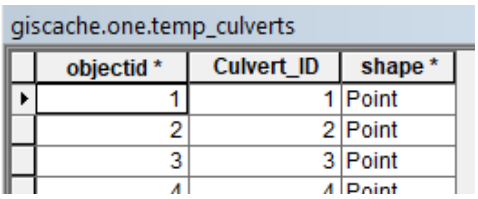
As an example, let’s say we want to change the schema of the culvert feature class, which has photos for many of the features. We first create a new feature class with the desired schema and enable attachments on this destination feature class. We then add a “Culvert_ID” field to both the origin and destination feature classes to act as our ID field. Next, we copy the individual features from the origin to the destination using an edit session and copy/paste, making sure the Culvert_ID field comes over cleanly. Finally, we run the tool to copy the attachments using this field. Afterwards, the Culvert_ID field could be deleted from either origin or destination if desired.



culverts_county

objectid *	Culvert_ID	cul_type	diameter	length
1	1	UNKNOWN	0	0
2	2	UNKNOWN	0	0
3	3	UNKNOWN	0	0
4	4	UNKNOWN	0	0

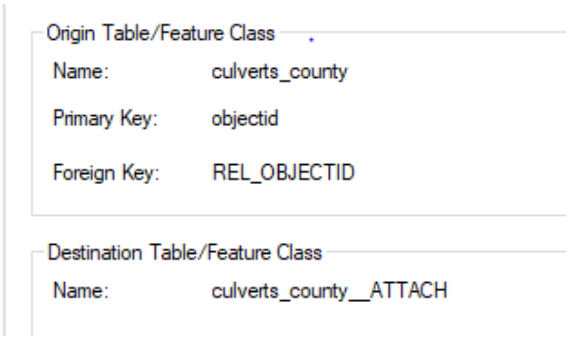
Source feature class with added match field (Culvert_ID)



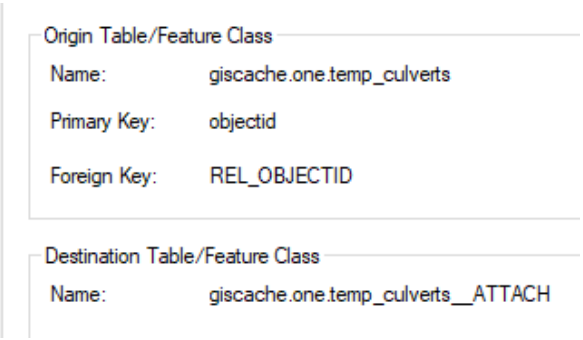
giscache.one.temp_culverts

objectid *	Culvert_ID	shape *
1	1	Point
2	2	Point
3	3	Point
4	4	Point

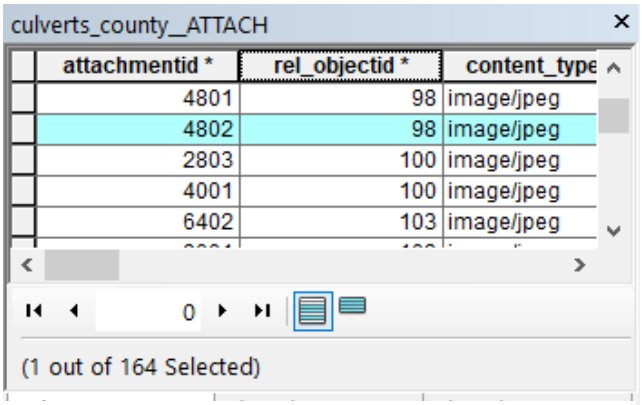
Destination feature class with match field



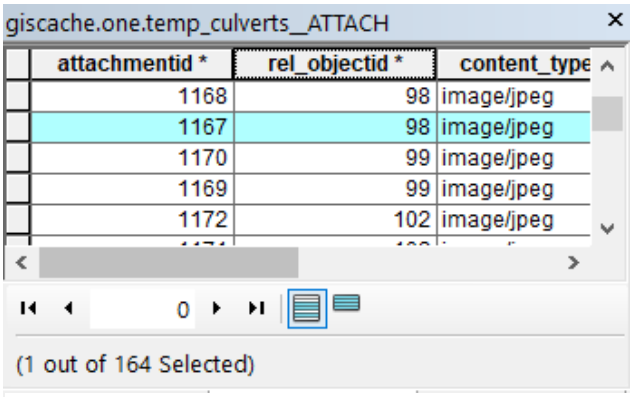
Source relationship class



Destination relationship class



Source attachment table



Destination attachment table. Note that the rel_objectid field does not match the source after 98 because the object IDs of the two feature classes do not match perfectly (though they are close in this example). However, because the match field was used, the attachments are on the right features.

ASSUMPTIONS

This script tool relies on the following assumptions.

1. Both feature classes have attachments enabled and participate in a standard ESRI attachment relationship.
    a. The associated attachment table has the same name as the feature class plus __ATTACH (if the feature class is giscache.one.flood_road_points, the attachment table is giscache.one.flood_road_points__ATTACH).
    b. The attachment table lives in the root of the SDE or GDB (for giscache.one/giscache.one.Transportation/giscache.one.flood_road_points, the table is giscache.one/giscache.one.flood_road_points__ATTACH).
    c. The feature class OBJECTID field is the primary key used to relate the attachment table to the feature class.
    d. The attachment table REL_OBJECTID, ATT_NAME, and DATA fields are the relate key, name of the attachment (“Photo1.jpg”), and actual data blob that is the attachment, respectively.
    e. The feature class and attachment table live within an SDE or GDB (which I think is a requirement for attachments anyways…).
2. The origin and destination feature classes have a common match ID field. It need not have the same name (ie, it could be “Culvert_ID” in the origin and “CID” in the destination) but must be the same type (integer, text, etc).
3. The match IDs must be unique in the origin feature class—a 1:1 relation between match ID and object ID. If there are many origin object ID's associated with one match ID, only the attachments of the last scanned object ID will be uploaded to the destination feature class. If there are many destination object ID's associated with one match ID (and one origin object ID), the same attachments will be added to each feature associated with that match ID (which could cause havoc if you have lots of attachments named “Photo1.jpg”). If there are many origin object ID’s and many destination object ID's associated with a match ID, the universe will implode… probably…

SCRIPT USAGE

As a geoprocessing tool/script, this script is meant to be run from within ArcGIS. The current version should live in the giscache.one.Transportation toolbox within the main SDE. It can be run from either ArcMap or ArcCatalog.

Preparation

This tool can be used as part of the process of copying features to a completely new feature class or for appending features to an existing feature class. Before running, make sure the following statements are true:

1. A destination feature class should exist.
2. The destination feature class should have attachments enabled.
3. Both the origin and destination feature classes should have a common match field containing the unique match IDs (see assumption 2).
4. The match IDs should be populated in the origin feature class. A quick and dirty way to do this is to copy the object ID into the match ID field with the field calculator.
5. The destination feature class should already contain the features that will be linked to the attachments. For a new destination feature class, this could mean copy and pasting the features from the origin feature class in an edit session (after populating the match ID field).

Once these conditions have been met, run the script tool from the Transportation toolbox. I've tried to include useful information in the tool's built-in help data so that you can use the "Show Help" button to get an explanation of each field. I've also included a brief rundown of each parameter here.

Origin Feature Class

The feature class that has the attachments you want to transfer to another feature class. The script automatically determines the attachment table path based on the name you choose here, so make sure you choose the right feature class.

Destination Feature Class

The feature class that has the features to which the attachments will be added. Again, the script automatically determines the attachment table path based on the name.

Origin and Destination Match Fields

The fields in the origin and destination feature classes that contain the unique match IDs for each feature.

Scratch Folder

A folder to store the downloaded attachments; if it does not already exist, it will be created. A folder will be created within this folder with the date and time (yyyymmdd-hhmmss) the script was run. Within this folder, a separate folder will be created for each origin feature that has attachments. These folders will be named after the origin object ID and will hold the attachments for that particular feature.

In addition, the script places add_table.csv here. This table are used by the add attachment tool to define which attachments go with which object ID. The first field is the destination feature class object IDs and the second field is the path to the attachments associated with that object ID.

You normally shouldn't need to delve deeply into the scratch folder stuff; it's there as a staging area for the data. The script doesn't delete it afterwards so that we have a trail of what happened for debugging purposes. If the script is run multiple times, this folder could get bloated.

SDE Username and Password
If the destination feature class resides in an SDE, we have to create a temporary version in order to edit the attachment tables and upload the attachments. This requires a username and password to the SDE. These will usually be your personal username/password that you used to set up your SDE connections.

The username and password fields can be left blank if the destination feature class is in a File GDB.

Script Parameters for Reference

Variable	Parameter #	Data Type
Origin Feature Class	0	Feature Class
Destination Feature Class	1	Feature Class
Origin Match Field	2	Field
Destination Match Field	3	Field
Scratch Folder	4	Folder
SDE Username (optional)	5	String
SDE Password (optional)	6	String (Hidden)

OUTPUT

Once you've chosen the right feature classes and fields, click start and away it goes. It took about a minute and a half on my test feature class (county_culverts), which had about 160 photo attachments in 85 features. Once it's done, you should be able to see the attachments in the destination feature class attachment table and with the identify tool.

ERRORS

This script does some basic error checking on the input parameters to make sure the feature classes and attachment tables exist, etc. If you get an error saying it can't find the destination attachment table, you may have forgotten to enable attachments on the destination feature class (I've forgotten to do that more than once).

Any errors should be displayed in the geoprocessing status window like any other tool, and if it fails it should show up in the history as a failed tool with the error messages attached like normal.