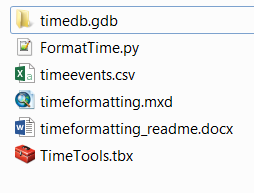
**Format Time Script**

This doc is accompanied by a csv file of data points that have incompatible separate time and date fields, along with a lat, long fields. The data has generic fieldnames and is in a fake location. The goal is to create a new time fields that works with the time slider tool in ArcMap, so each point can be shown. The project makes use of logic that does common things with time fields. The key is a python script that can reformat any time formatting into a usable format.

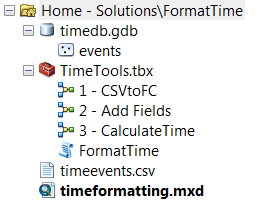
Unzip the project and place it in this folder – create it if it doesn’t exist.

C:\Solutions\FormatTime

I am using 10.4 and have python installed using the default install. This is what is in the project:



Open TimeFormatting.mxd in ArcMap. If you look at the content in the catalog window it looks like this expanded.



This is done with a series of three GP models. The last one uses a python script as tool within the model. (You can’t see the python scripts in the catalog window but you can see them in windows explorer).

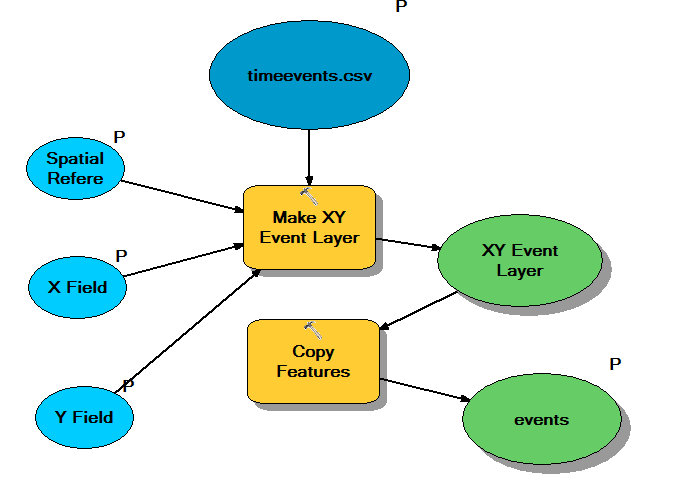
Each model is run from 1- 3 and is discussed below. The first three could be done manually instead. They are very simple. The last one uses the two script tools. The only input data is the timeevents.csv.

**1 - CSV to Feature class Model**

This model replicates the following manual steps so you don’t need to do them:

* In the catalog window create a new empty file geodatabase ( rt click and choose new /file geodatabase in the catalog window).
* Drag the csv to the table of contents (TOC), and export the csv file to it (rt click on cvs and choose export and pick the fgdb you made).
* From the fgdb, add this new table to the TOC and remove the original one from the TOC.
* Choose display X Y data (rt click on it and pick XY data) and pick WGS84 coordinate system in the wizard screen and the lat long fields. This adds a new layer to the TOC as an EXY event layer, but it still isn’t a featureclass.
* Export the resulting XY Event layer to a featureclass (rt click and export).
* Remove all the previous layers from the TOC except for this new featureclass.

Here is the model:

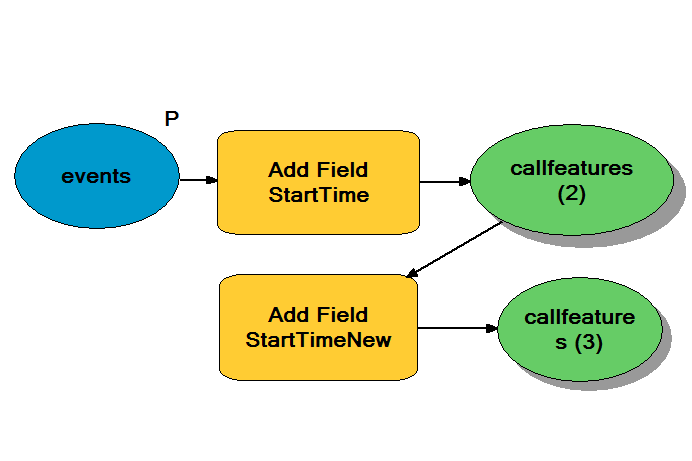


**2 - Add new fields to use for calculations:**

This very simple model replicates the following manual steps:

* Add the following new text fields that our python scripts will use: StartTime, StartTimeNew

Here is the model:



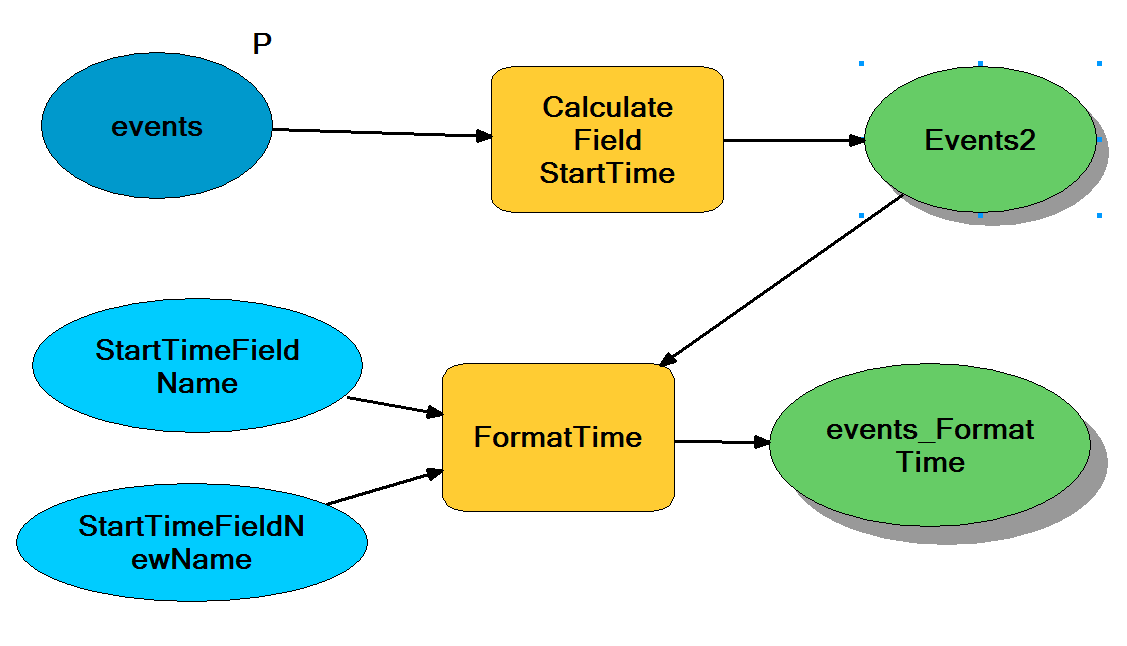
**3 - Calculate time fields:**

The final model does the real work in calculating fields.

* StartTime – The incorrectly formatted time except now in one field.
* StartTimeNew – The correctly formatted time.

Python scripts are used for this since the out of the box format time tool unfortunately doesn’t support the input time format to then translate it to a new StartTimeNew.

Here it is:



**This first step in the model calculates the StartTime Field:**

This can also be done using these manual steps:

The problem is that our time is in a format not supported by ArcGIS for the time slider tool. The time is in M/D/YYYY H:MM:SS AM and the ArcGIS time slider tool needs the format YYYY/MM/DD HH:MM:SS.s (the last s means an AM or a PM). Also, we need the time in one field first to convert it. First, open the attribute table and use the calculate value function for the StartTime field (right click on the field name and choose calculate values) and then use this expression using the two fields separated with a space ([DAY] & " " & [TIME\_] ) . This puts the date time as one temporal field. It is still not OK for use with the time slider tool though.

**The second step calculates the StartTimeNew field:**

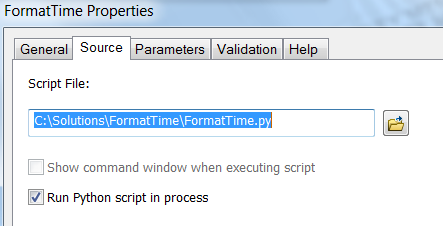
The out of the box tool in ArcMap can read and reformat many time formats, but not the format we are using, ie M/D/YYYY H:MM:SS AM. We need a YYYY/MM/DD HH:MM:SS.s format. The model uses a script tool.



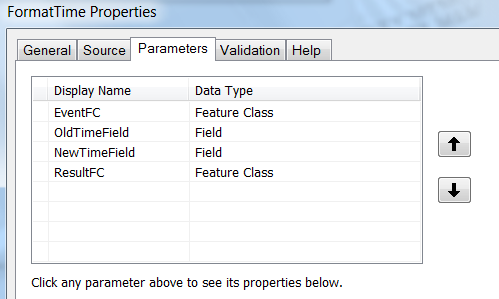
If you put the project in the same path as I did this should work. If not you will need to edit the script tool to make sure it is pointing to the correct python script location. A script tool does two things:

It is a special tool that allows a custom python script to be used in a model as a GP tool within the model. It points to a python script. Right click on the script tool in the catalog window, and go to properties. Make sure the path is correct to the python script.

*See Advanced Notes at the end to explain more about to edit the script to handle ANY time formatting you might encounter*.



Script tools can be configured to have parameters, both input and output, so they can be linked to inputs when placed in modelbuilder.



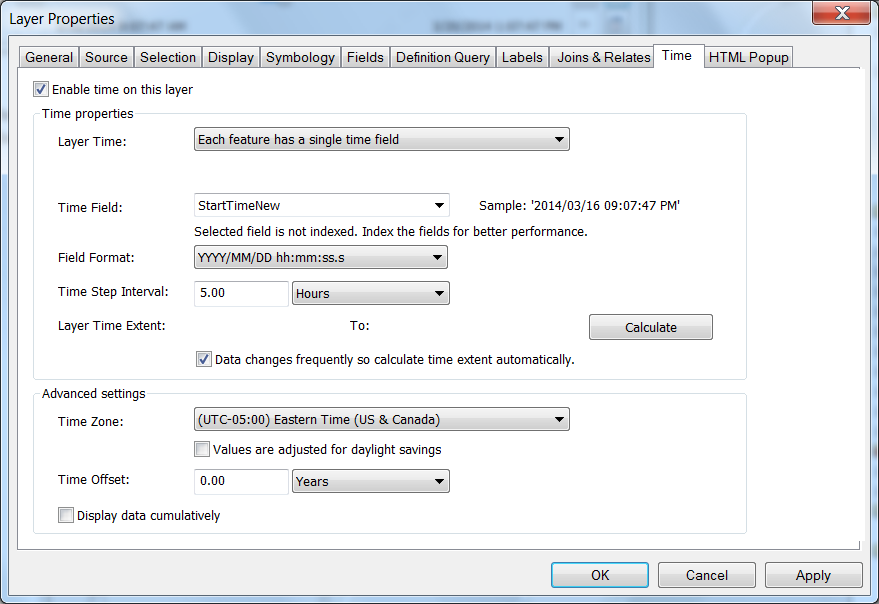
**Running the Models:**

The project should work right away if you unzip it and place it into this folder structure that I showed at the start of this doc. Run each model in sequence.

**Now we can use the time slider tool:**

There are a couple gotchas and three places to set time: the layer, the tool, and the tool options.

For the layer go to time properties and use these settings:

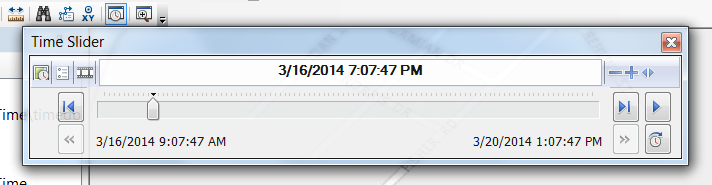


A couple notes:

* Be certain to use our StartTimeNew Field
* The field format ends with ‘ss.s’
* Click the calculate button to get the full data range calculated.
* Always choose a timezone!!! This is a critical step.

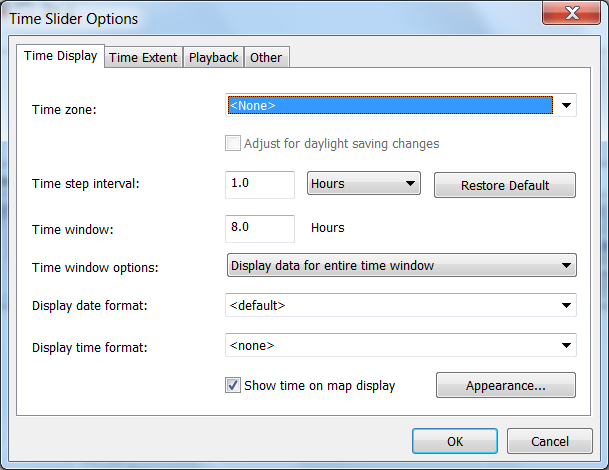
I saved the doc with “Display data cumulatively” unchecked, because I wanted to add a time window later. Uncheck or check this as you like.

To bring up the time toolbar go to the little clock icon on the main toolbar and click it to bring up the time slider tool.

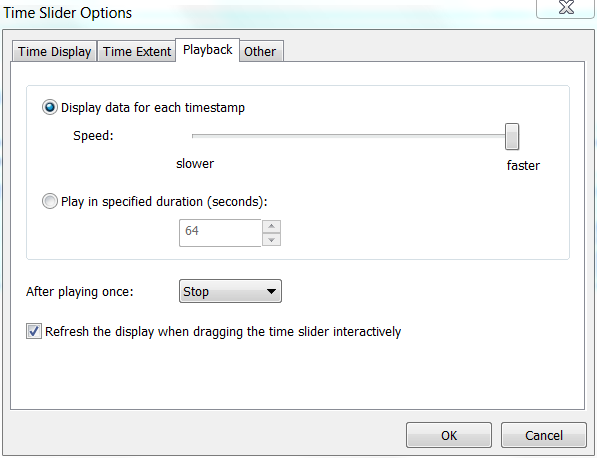


Then enable time by clicking the clock on the time slider toolbar itself in the upper left.

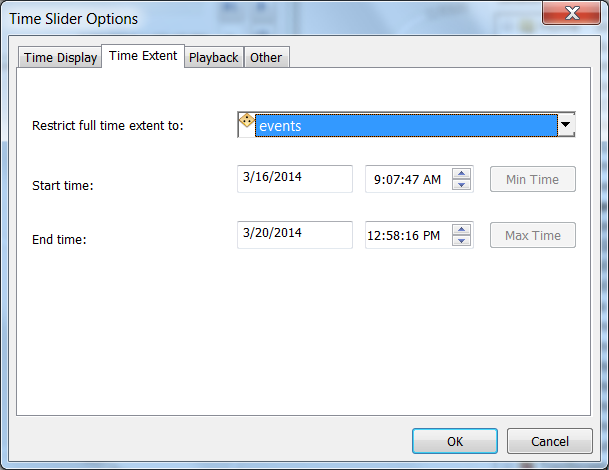
It won’t look like this yet. Click the options button on the time slider window in the upper left and set the options to this. Choose a time window like 8 hours.



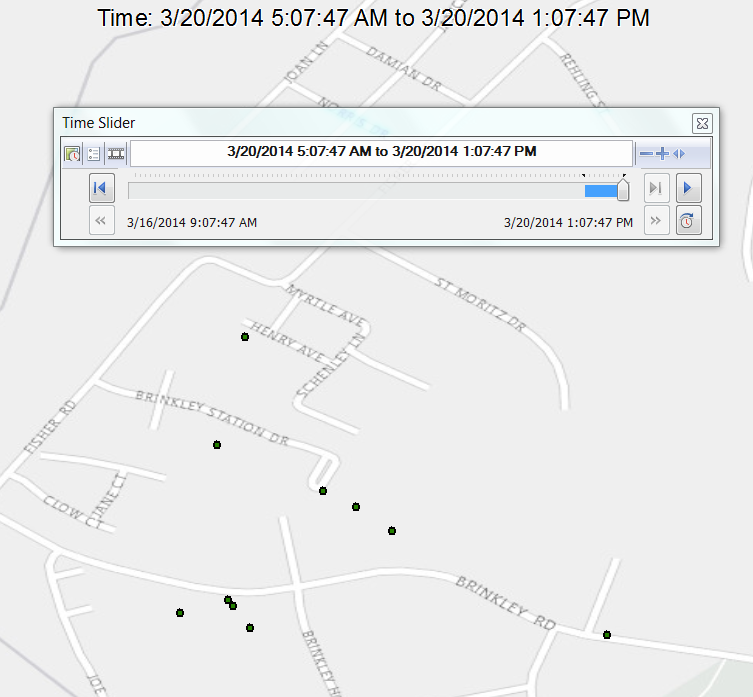
Set these options too:



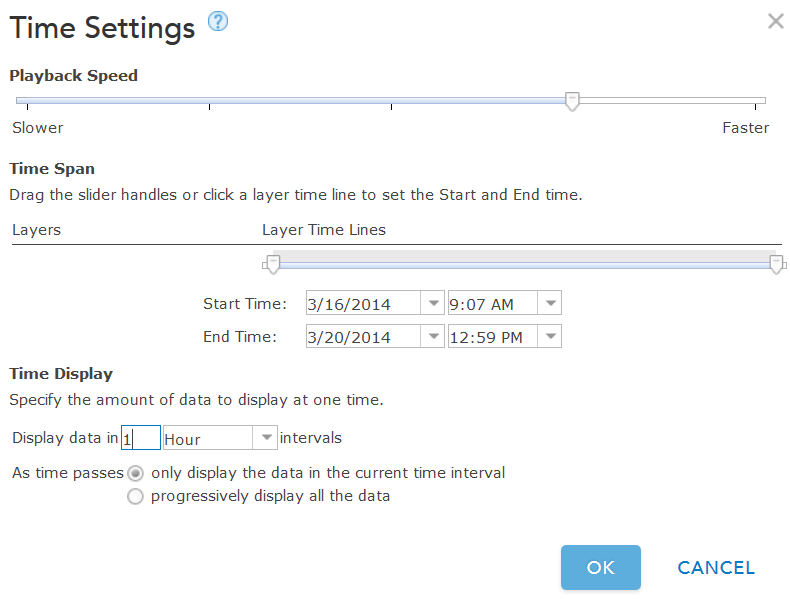
And this:



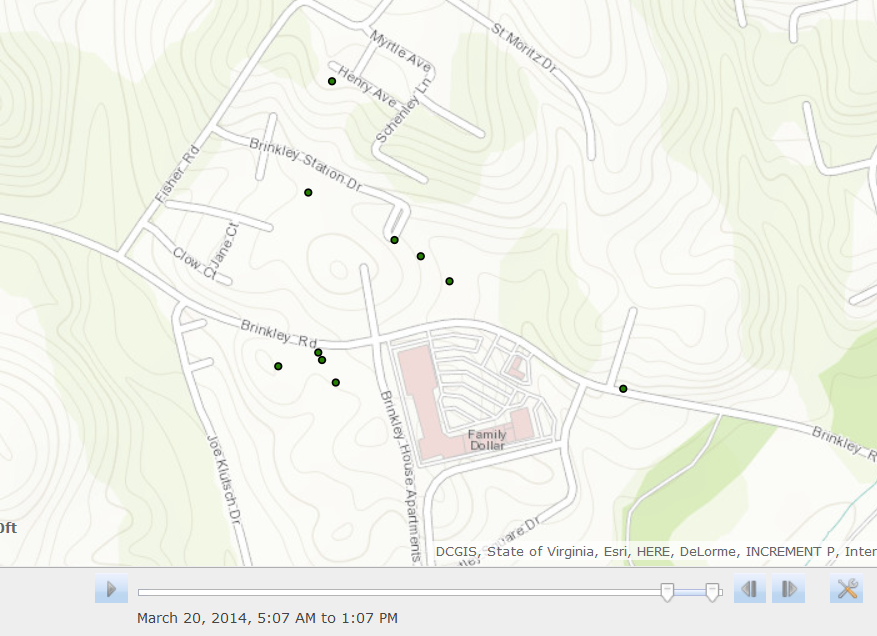
The result is this:



If you want to publish, remove your basemaps, give the map a title and tags, and share to ArGIS Online or Portal. You actually are publishing a hosted feature layer. Add this to a new webmap, click on the layer and choose more/enable time animation. On the time toolbar that appears click the gear to get to the advanced settings:



Set to 1 hour increments. Note that our data was not shifted because we entered a time zone.



**Advanced Notes**

The python script looks like this:

# FormatTime.py

# This tool reformats a single datetime field from M/DD/YYYY HH:MM:SS AM format to YYYY/MM/DD HH:MM:SS AM for a featureclass

# Reads a string from a field called "StartTime" (text) in a M/DD/YYYY HH:MM:SS AM format and

# writes to an different existing field called "StartTimeNew" (text) in a YYYY/MM/DD HH:MM:SS AM format

# The time slider tool needs the date in this format, thus this script

# To use

# - make sure your date field is in the format M/DD/YYYY in one field and your time field is in a separate field in HH:MM:SS AM

# - you want these to be in one field together

# - Add a new field called StartTime (text) to your featureclass

# - If not in one field, using the field calculator in ArcGIS, append your date and time using [date field] & " " & {time field}

# - Add a new field called StartTimeNew (text) to your featureclass to hold the new calculation that this script will perform

# - edit the path below to the feature class (not table) that contains the above fields

# - run in the python program that is installed standalone with ArcGIS

# - Line 39 and 41 control the input and output formatting, you could change this any time format

import arcpy, os

from datetime import datetime

# Input variables

# featureclass

pFeatureClass = arcpy.GetParameterAsText(0)

#parameters for field names

pStartTimeField = arcpy.GetParameterAsText(1)

pStartTimeFieldNew = arcpy.GetParameterAsText(2)

rows = arcpy.UpdateCursor(pFeatureClass)

print("start")

for row in rows:

#print(row.getValue("StartTime"))

StartDateObject = datetime.strptime(row.getValue(pStartTimeField), "%m/%d/%Y %I:%M:%S %p")

# ie M/DD/YYYY HH:MM:SS AM

newStartDateStr = StartDateObject.strftime("%Y/%m/%d %I:%M:%S %p")

# ie YYYY/MM/DD HH:MM:SS AM

row.setValue(pStartTimeFieldNew, newStartDateStr)

#print(newdateStr)

rows.updateRow(row)

# Delete cursor and row objects to remove locks on the data

del row

del rows

print("done")

Note that # are comment lines and don’t do anything. The key two lines are the two in red. The first one uses a shorthand notation for describing the current format of the datetime field. The second one uses a similar notation to describe the desired formatting. All you would need to do to use this script with other types of formats would be to :

1. Understand the shorthand
2. Edit and save the script

To understand the shorthand see the python documentation on time nomenclature at:

<https://docs.python.org/2/library/time.html>

The other part of the script simply sets up the two timefields and the featureclass input. The order of the parameters matter 0,1,2 as they must match the order in the script tool that points at this script.

For instance:

pFeatureClass = arcpy.GetParameterAsText(0) # featureclass

#parameters for field names

pStartTimeField = arcpy.GetParameterAsText(1)

pStartTimeFieldNew = arcpy.GetParameterAsText(2)

Corresponds to the order in the first three input field. (The last is an output.)

