SHC_DataSpider: Release 1.0

> "On two occasions I have been asked, 'Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?' I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question."
>
> -*Charles Babbage* , on the difference engine

History:

2011-12-07/8	DVD	Draft Document; ready Beta1 Release
2011-12-14	DVD	Updates, include pGDB & Feature Dataset Introspection
2012-01-17	DVD	Schema Info
2012-02-23	DVD	Problem Data Section
2012-04-17/18	DVD	Updates for Release v.1.0
2014-04-18	JS	Updates for Release v.1.0

Table of Contents

Summary:

This document details the Beta1 release of the Klamath Strategic Habitat Conservation Program data inventory tool, *SHC_DataSpider*. This release is packaged as an ArcGIS 10/10.1 toolbox tool only, distributed in `.tbx` format.

The tool inventories a file hierarchy beneath a starting location. It creates an extent Feature Class (*ExtentFC_WGS84* by default), which details the extent rectangle for every spatial data set with a known coordinate system in GCS WGS84. In addition, it creates a number of tables:

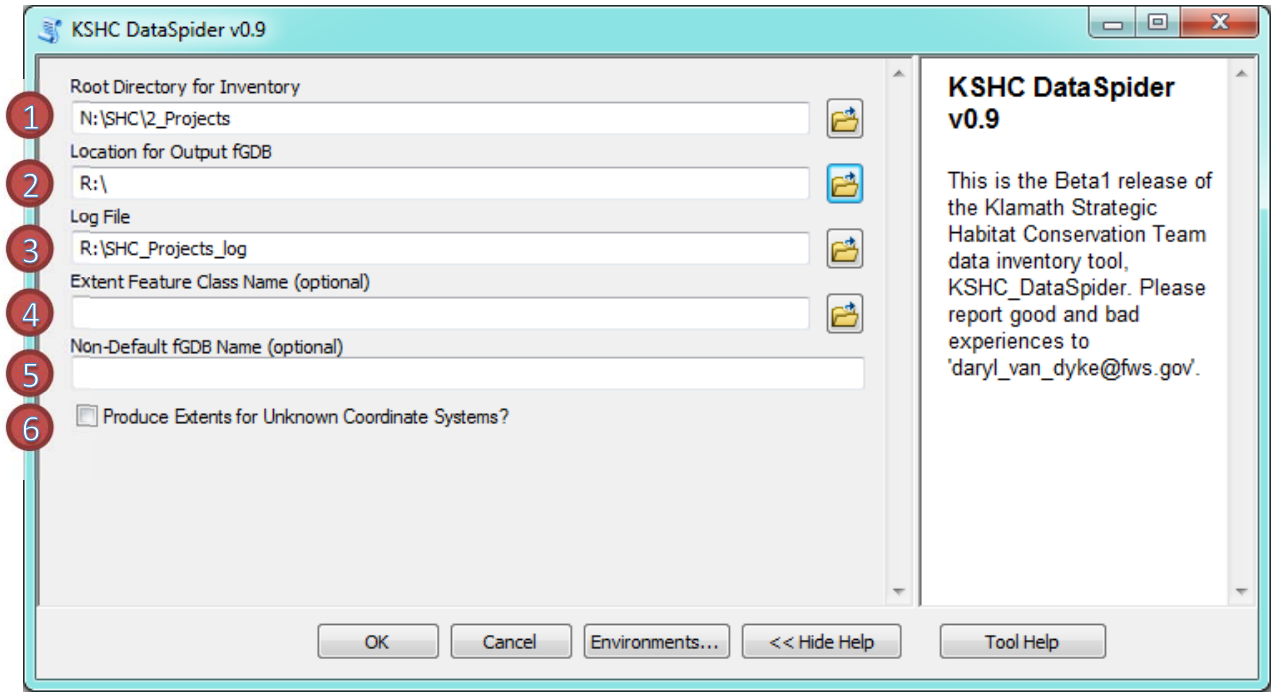
- *tblExtentErrors* – Data sets with missing coordinate system, or bad extent objects

- *tblErrors* – Data sets that misbehave when queried by ArcGIS (e.g. bad Describe objects)
- *tblRasters* – All raster data sets
- *tblTables* – All tabular data sets
- *tblVectors* – All vector data sets
- *tblWorkspaces* – A list of all unique workspaces
- *tblWorkspaceTypes* – A list of all workspaces, by type. A folder of shapefiles and layer files will be listed here twice, once as layer folder, once as a shapefile folder.

Please, do let me know what your results are. If the SHC_DataSpider really helps, let us know. An accurate assessment of the value of the product helps us. If something crashes, please send me the log file (as a text attachment) as well as a screen shot (alt-print screen, then paste) of the ArcCatalog progress dialog for the tool.

Configuration:

The configuration should look as follows:



1. **Root Directory for Inventory**

This is the root of the search tree for the inventory. A complete inventory of a corporate GIS can take days, and may be memory limited. It is gently suggested that inventories be done of sub-sets of data, at first, or a test project directory.

2. **Location for Output fGDB**

This denotes the *folder* that will contain the generated fGDB.

3. **Log File**

This will the prefix for a file generated with *<input>_<timestamp>.txt* as an output. The timestamp is YYYYMMDDHHmm, and will be common to both the name of the fGDB and the associated log file.

4. **Extent Feature Class Name (optional)**

This provides the option to force the use of a non-default name for the extent feature class.

5. **Non-Default fGDB Name (optional)**

This provides then option to force the use of a non-default name for the resulting file geodatabase.

6. **Produce Extents for Unknown Coordinate Systems?**

This option is left UNCHECKED by default. When producing extent rectangles for geospatial data, *KSHC_DataSpider* will attempt to re-project on the fly into GCS WGS84. This operation depends on having the coordinate system information set for the target data set. If this is not the case, the tool will write the data set to the appropriate table (*tblVectors* or *tblRasters*) and note the CoordinateSystem filed as 'Unknown'.

If this option IS selected, the resulting extent rectangle will be incorrect. For example, a UTM NAD83 data set in the Klamath Basin will be located around (400 000, 4 500 000) in lat/long. This is obviously wrong, and ArcGIS will complain when you view the extent FC. Further, it will make the extent rectangles very small when viewed in preview mode – the extent of the extent FC is now very big. The purpose of this option is to provide a way of assessing the spatial patterns of geospatial data with an 'Unknown' coordinate system. Because 'real' values of lat/long will be constrained by the actual number of degrees, any feature extents with large values (> 180) are really a form of projected data. UTM NAD83 data will cluster in on 'area', and State Plane data (in US Feet) will cluster in another.

Usage Notes:

Memory Issues

As with most software development, the end of the first cycle is the beginning of the next.

Currently, the software is `list` driven. This means that the actions for the inventory process are executed sequentially:

1. Model the file hierarchy as a tree
2. Inventory all directories on that tree for data

3. Process the discovered data into tables and a feature class

This may change in future releases; there are benefits and drawbacks to this model. One drawback is that the program is RAM-intensive for large data structures, which means that memory can be a limiting factor when trying to create a large inventory.

As a result, for very large inventories, memory can be an issue. Use the process manager to monitor the size of the memory allocation taken up by the tool. If it starts to grow above 2 GB or so, I'd recommend breaking the inventory into the next level of the hierarchy. Instead of running it on your entire `N:\\`, for instance, do `N:\\agency`, then `N:\\raster` etc.

The above assumes a 64 bit operating system. If you have a 32 bit OS (like Windows XP), then you have to share less than 4GB of RAM (assuming you have that much installed) between OS, ArcGIS, and the inventory tool. Either way – use the process monitor/task manager to monitor RAM consumption.

GUIDs and Database Joins/Relates

The *KSHC_DataSpider* was designed to support complex database analysis of GIS data structures. To that end, the tool generates a number of tables, and associated unique key values. The data tables are *partially* normalized by design; there are number of features (additional layer file functionality, and MXD introspection, for instance) which have not been implemented. The table structure was designed to be forward-thinking in this capacity.

All of the generated output include GUIDs for database joins and relates. While the geospatial data table (attribute) structure is not completely normalized, the raster and vector table attributes are different. In order to view the full set of attributes for data sets, create a join between the *ExtentFC_WGS84* and *tblRasters* or *tblVectors.* Likewise, a one-to-many relationship like 'data objects in workspace' can be modeled with the supplied GUIDs. Finally, although ugly, these GUIDs make the product 'version-friendly' for SDE; this motivates the use of GUIDs (geodatabase-managed GlobalIDs) for keys as opposed to the paths themselves, either as strings or hash values.

Join	Source Field	Target Field
tblVectors on to *ExtentFC*	ExtentFC.GUID_VectRas	tblVectors.GlobalID
tblRasters on to *ExtentFC*	ExtentFC.GUID_VectRas	tblRaster.GlobalID

Other Notes

- **Is a File Raster considered a folder? Is a Geodatabase considered a folder?**

The ESRI File Raster is a data model that can be confusing: Like a fGDB, it 'looks' like an ordinary folder from the perspective of the OS file system. When viewed in ArcCatalog, the File Raster and fGDB objects appear as collapsible containers that hold familiar geospatial files.

We have probably all viewed a file geodatabase or a raster from windows explorer. Worse – we might have made the mistake of accidentally dragging and dropping into those folders. KSHC_DataSpider includes logic to handle these situations correctly. Examine the errors in the log file. A drag-and-drop error will show up as '…/some.gdb/a_folder/another.gdb couldn't be read' if a file data container has been moved inside of another container.

Indicators of Problem Data/Objects (or; not a bug, it's a feature):

Basic Troubleshooting

Generally, there are four scenarios you will encounter:

- SHC_DataSpider crashes, and gives me an "Exception on line XXX" message. This message indicates a problem with the python code, and should be rare in occurrence (please send me a log file or screen shot).
- SHC_DataSpider crashes, and doesn't report an exception value. This indicates a problem at a deeper level; usually, a corrupt data set or MXD. For help on this topic, refer to the section "Diagnosing Bad MXDs".
- SHC_DataSpider crashes, and returns an "out of index" error. This is a good indicator that you are trying to inventory too much. It's probably an error in the ArcGIS components used to create this program.
- ArcCatalog terminates and the OS throws up a dialog box about an error with the 'C… library'. This problem has to do with how the arcpy elements behave , and may or may not be 'my fault', in the sense that I can do anything about it. It would be impossible to design a tool that could handle all 'broken' or corrupted data.

If the tool reproducibly crashes at the same data element or data set (table or FC), then try moving or zipping the file and re-running. En example of this diagnostic procedure is included below. If the data can be viewed through ArcCatalog (in preview tab) it will *probably* play nicely with the KSHC_DataSpider.

For more information, see the "Indicators…" section at the end of the document.

When in doubt, re-run.

Crashes at a very low level are indicators of problem data. The crash message:

- Undefined signature file type

  - This was due to a Bad MXD; refer to the "Diagnosing Bad MXDs" section below.

- `Fatal Python Error: Deallocating None`
  - Problem with the object reference management of memory arrays; probably due to a problem in the ArcObjects underlying the `arcpy` wrapper. I've only encountered this error when traversing a directory of many (> 1000) rasters. The `Deallocating none` error, in Python, occurs when an index or counter gets out of sync with the actual memory stack. Try re-running on with a smaller set of rasters in the target directory.

Diagnosing Bad MXDs:

Bad MXDs happen. Here's an example how to locate them.

Problem: SHC_DataSpider is crashing while inventorying an MXD. First, we look at the output to determine which layer inside the MXD is causing the crash. This will be the last entry in the `tblMXDLayers` table. Looking in Fig. 1 below, we can see that it managed to inventory all of `Mosaic.mxd` just fine.

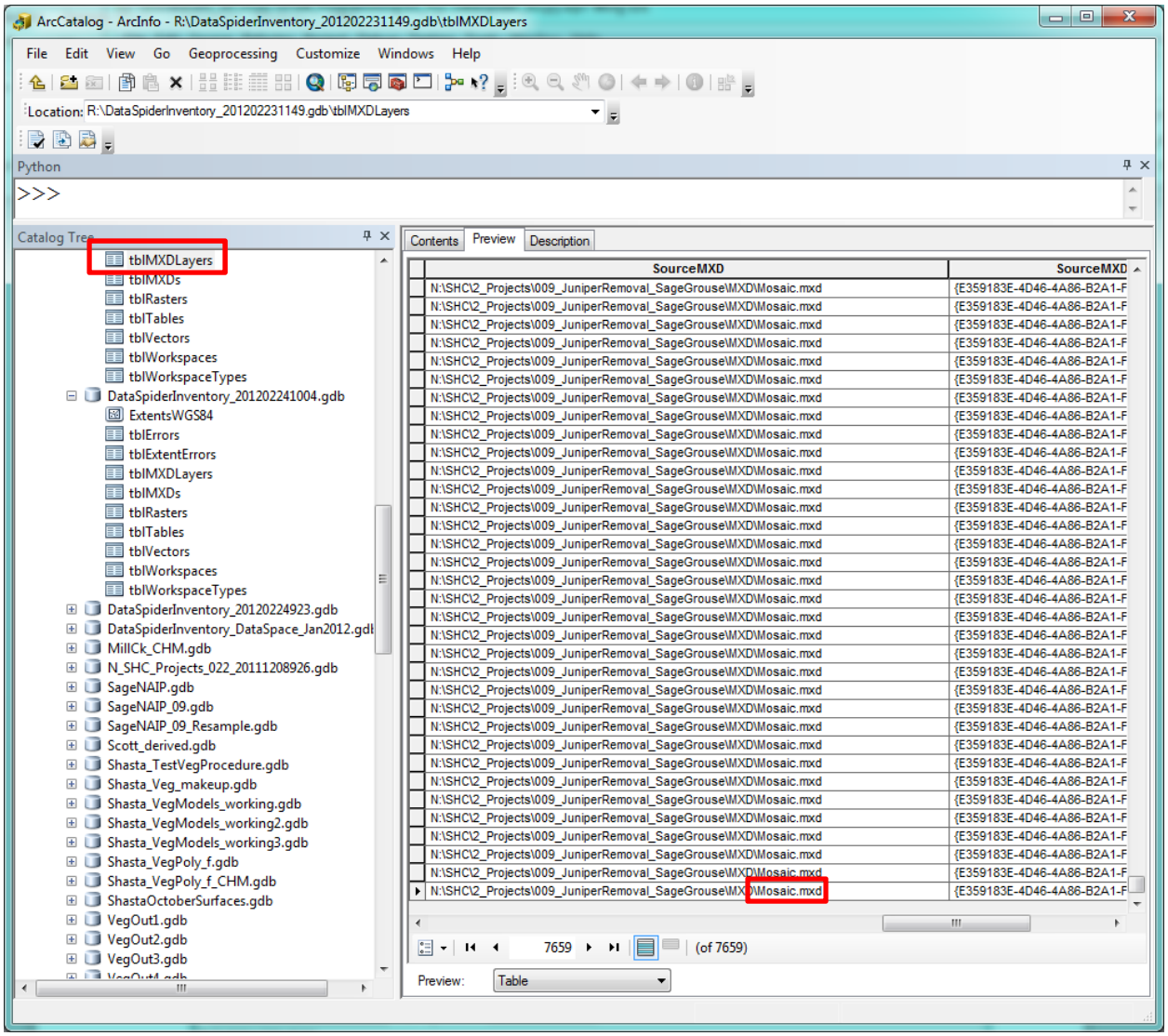


Figure 1 - After an MXD crashes the SHC_DataSpider, look inside tblMXDLayers to identify the last layer that was successfully inventoried.

Now we look at tblMXDs (Fig. 2). We can see that right after the SHC_DataSpider finishes the `Mosaic.mxd`, it's going to try `Mosaic2.mxd`. Could `Mosaic2.mxd` be the problem?

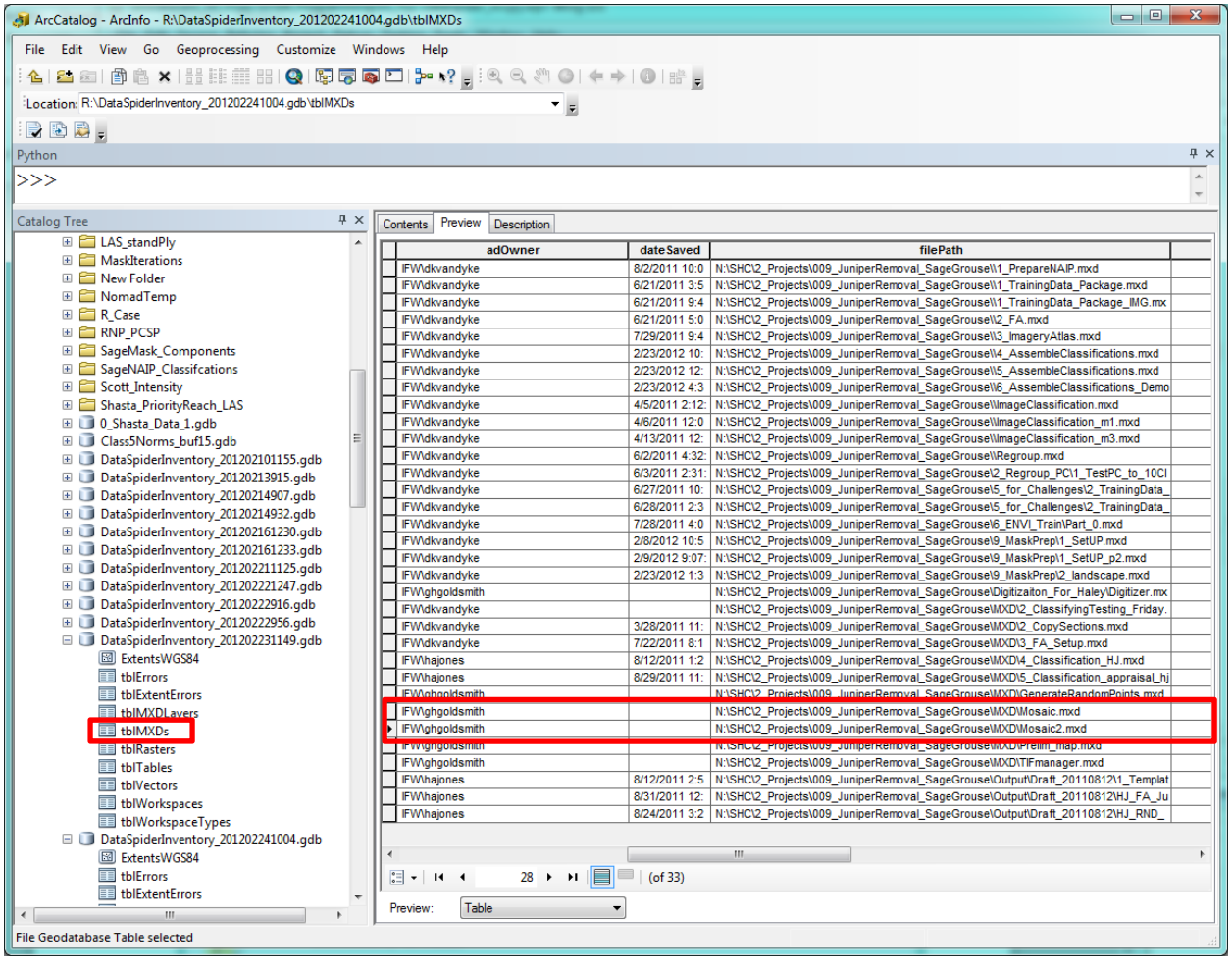


Figure 2 - After you've identified the last MXD that was successfully inventoried (Fig. 1), use tblMXDs to find the next MXD to be processed. This file is likely corrupted.

Now that we think we know the bad MXD document, let's rename it with a ".BAD" at the end so SHC_DataSpider will ignore it on the next try (Fig. 3).

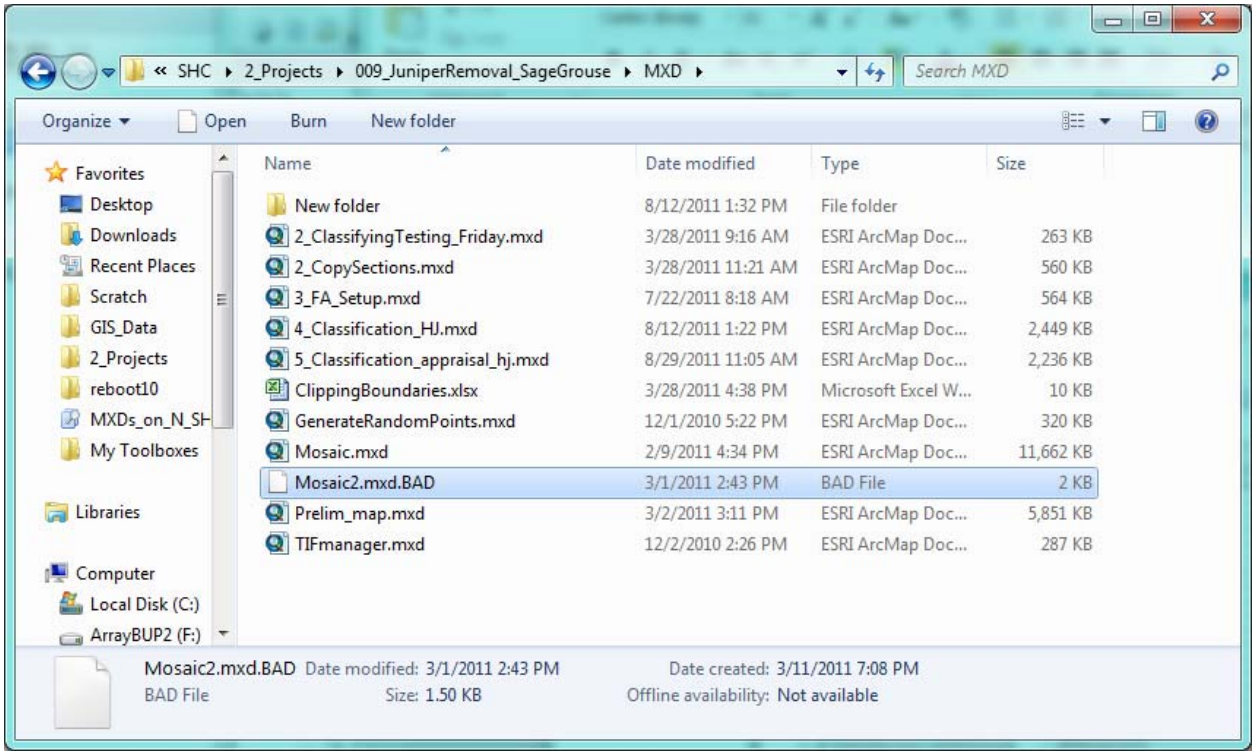


Figure 3 - Change the name of the corrupt MXD

Adding the suffix ".BAD" to an MXD file that is crashing the SHC_DataSpider will allow the file to be ignored in the next run. Note that this fix will only succeed if the FULL file name + extension is edited (e.g. `MXDName.mxd` → `MXDName.mxd.BAD`). If the file extension is hidden, changing the name will not fix the problem (e.g. `MXDName` → `MXDName.BAD`). **You will need to disable the 'hide extension of known file types' option in the Windows Explorer interface in order to accomplish this.**

Now we can try to run the SHC_DataSpider again. It worked!

Contact Info:

Daryl Van Dyke

GIS Analyst

U.S. Fish & Wildlife Service - AFWO

Klamath Strategic Habitat Conservation

daryl_van_dyke@fws.gov