Requirements:

* The code shall be a function that can be applied to every data transfer section made.
* The code shall report shall check the field names listed in the parameters to see if they are listed in the specified field names.
* If a field name(s) is missing from the parameter, the code shall add text to the report stating a list of all missing field names.
* If a parameter was left blank by the user, the code shall add text to the report stating the parameter that was left blank.
* The final report produced by the code shall include a brief set of instructions for each individual issue so the user may have a better understanding of what went wrong and how to fix it.
* The final report produced by the code shall be clearly organized into clearly marked sections each detailing the issues that arose when converting each table of data.

New lines of code:

* 66-81: These lines create a new folder where the csvs will be stored so they aren’t mixed in with the status report. This way the user won’t have to remove the report from the csvs before uploading them into ICPR4.
* I’ve also modified the beginning of every section of code so that each csv is now saved in the designated csv folder.
* 149, 155, 161: I’ve added test messages to “Finally” and “Except” statements at the end of the Node section. These were only added to test which branch the program went down under various conditions. I suggest leaving them in there if you do your own testing and perhaps even adding them to the Links and Pipes sections.

I did some testing to see what errors showed up when the code ran under different conditions. The idea is that if you know what causes what lines of code to run (and what makes errors show up) will let you know when certain issues arise and what information needs to be written into the report.

|  |  |
| --- | --- |
| Condition | Results |
| No input for Node parameter | Code under Except Runtime and finally ran. |
| Correct input for Node parameter (Feature class for Nodes). | Code under Try and Finally ran. |
| Incorrect input for Node parameter (Feature class for Links) | Code under Try, Except Runtime, and finally ran. |
| Incorrect input for Node parameter (A random string) | Program did not run at all. Apparently it will only run if the field is either empty or has a parameter filled with the expected data type. So this is one problem you should not have to worry about. |

In summary, the code in “Try” will run so long as you put some kind of input in the Node parameter. The code under “Except Runtime” will run if the program is not given the information it needs to fill out the Node table and the “Finally” code will always run no matter what you do. Interestingly, “Except Attribute” never seems to run, at least I couldn’t think of a scenario in which it would. Still, probably best to leave it be.

Final thoughts and design ideas:

Again, I recommend using some loops to compare the array of all the field names in the ArcGIS table with those that are being searched for. Maybe there is already a predefined function that does this?

As for when to write stuff into the report, I recommend call the report function under the “Finally” block of code at the end of each data section. This part always runs after each data transfer section so this way it will always write something.

As for what will write though, that depends on the scenario. But I’ve got an idea for that too. Perhaps you can make two Boolean variables that serve as “flags”, one that switches TRUE when “Try” runs and another that switches “TRUE when “Except Runtime” runs. You can then have the function take these as parameters and use them to determine what errors arose during the data transfer and print out messages accordingly.

For example, say the user inputs an acceptable Node feature class with all the required field names. The Try flag would be marked as “TRUE” while the Except Runtime would remain “FALSE”. The function would see this and then know everything went perfectly and there are no issues to report.