# Checklist for Migrating Existing Scripts to 3.0 Framework.

This process assumes that the current scripts are adhering to the variable branching best practice. For scripts that are not using variable branching, a more manual approach is required.

1. Obtain the Source Data. There are two options. Data Manager, and Excel Export:

**Data Manager**

* + Ensure that all Standard Choices that contain EMSE scripts are marked as “EMSE”.
  + From the source agency, request a data manager extract of the following:
    - AA Configuration – Event Manager - Events (All)
    - AA Configuration – Event Manager - Scripts (All)
    - AA Configuration – Standard Choices (EMSE)
  + Create a temporary agency in an environment (avsolarc, av.supp, etc.)
  + Import the data manager package into the temporary agency

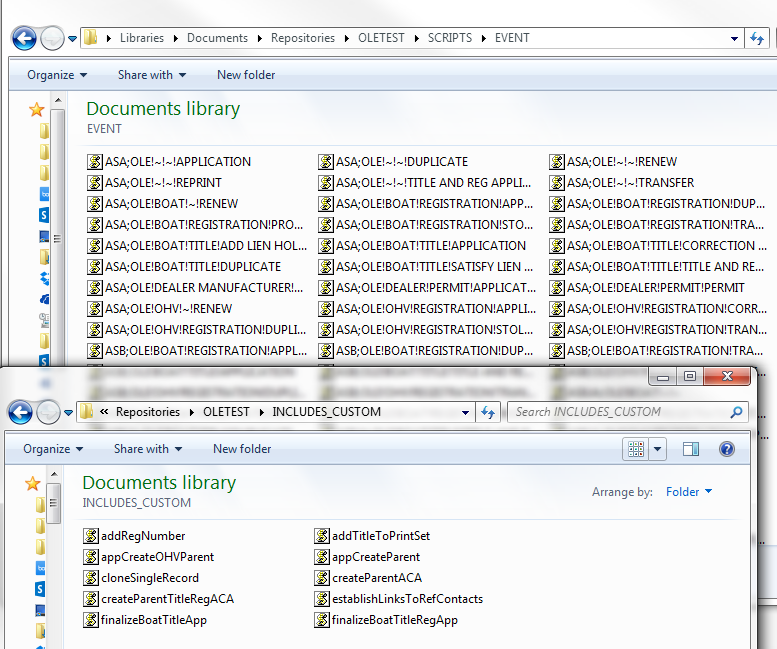
**Excel Export**

* Ask the agency for the following tables in Excel format
  + REVT\_AGENCY\_EVENT (event to script mappings)
  + REVT\_AGENCY\_SCRIPT (event scripts)
  + RBIZDOMAIN (standard choices)
  + RBIZDOMAIN\_VALUE (standard choice values)
* Open the RBIZDOMAIN worksheet
* Filter on the agency code and REC\_STATUS = “I”
* Open the RBIZDOMAIN\_VALUE worksheet
* Disable (set to I) the RBIZDOMAIN\_VALUE entries for any disabled standard choice (REC\_STATUS is I in RBIZDOMAIN sheet). Note these in the release notes.
* Filter by the proper Agency Code
* Filter by a caret (^) in the VALUE\_DESC column. This will include only valid script entries
* Sort by BIZDOMAIN and BIZDOMAIN value ascending
* Use find/replace to replace all double quotes with single quotes
* For the BIZDOMAIN\_VALUE column, use the custom text format of \”A@\” or, rename the column and create a new BIZDOMAIN\_VALUE column that has a formula of =CONCATENATE("A",TEXT(C5874,"000")) NOTE: We need to get this column to convert properly to the right sort order. This may take some experimentation, make sure you check the result in the next step. If this column is converted as an integer you will have problems.
* ~~Filter out Disabled Standard Choices (REC\_STATUS of “I”) – IMPORTANT!~~
* Copy the filtered data to the clipboard, then use the following page: <http://www.csvjson.com/csv2json> to convert it to JSON
* ~~Copy the JSON that is output into the top of the “load standard choices from JSON.js” file in misc folder of the EMSE Master Script Distribution.~~

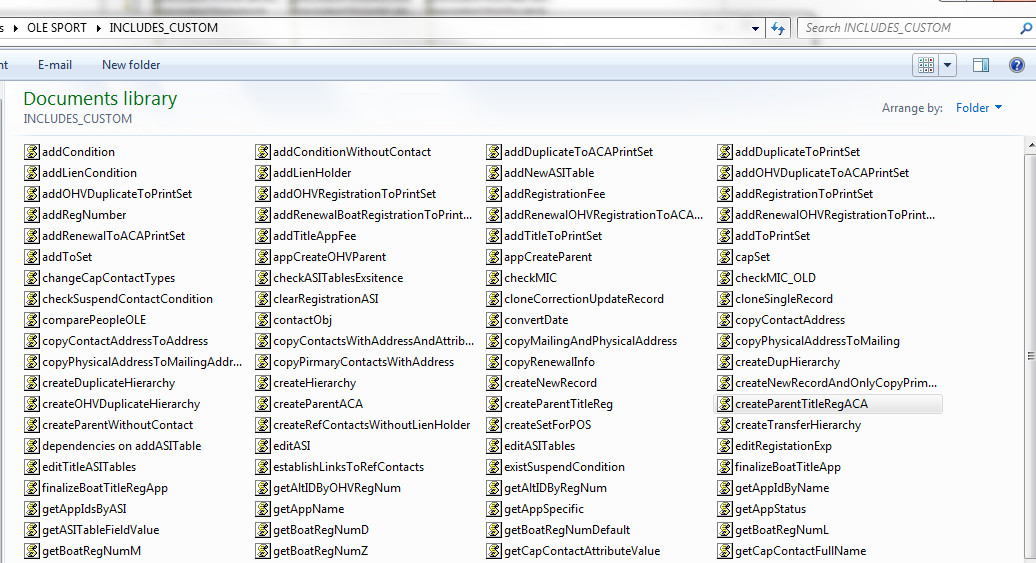
1. Copy the "Bizdomain to Script Converter – Delimited - NODE” to your conversion directory (It’s in the Misc directory of the Scripting 3.0 distribution). Copy the JSON above into the getObject function at the bottom. Run the script using Node.js on your local computer as shown.

Node “Bizdomain to Script Converter – Delimited – NODE.js” > scriptout.txt

1. Save the Script Output as a text file in your working directory (e.g. scriptout.txt).
2. Convert all LF to CRLF.
3. Import this file using Access text file import. Import it into the existing Scripts table structure. Caret delimited (^), backward quote (`) as text qualifier.
4. You should now have a populated Scripts table. Use the “Scripts” Form to review the Scripts table, looking for bad code, stuff that shouldn’t been converted, EMSE Global Flags, script entry points (e.g., ApplicationSubmitAfter) that aren’t required in 3.0, etc. Mark these as “No” in the Convert? Column. Make sure that any good and necessary code in these scripts are moved to appropriate spots. You can create new scripts if you need to at this point.
5. Check for any standard choices that were disabled. These will need to be either marked as convert? = no, or the code within them commented out, depending on what the customer would like to see.
6. Run the “CountAppearances” Macro. This will populate the number of times a branch is called from other code. A good rule of thumb is to create a function for any branch called more than once. The CountAppearances macro will do this for you by marking Function? = true and supplying a functi on name. Check th e function names and make sure they look good for your agency.
7. Back up the Script Table by doing a copy/paste. Do this often. You’ll thank me.
8. Run the “Replace Branches (View)” query. This will take all the branches that are not marked as functions and insert their text into the calling code. If the results look good, run the Replace Branches (Update) query. There are “1” and “2” versions of this query. Run it multiple times in order to process nested code. When zero rows are updated you are done.
9. Run the “Set Convert Flag to False for Embedded Branches”. We don’t need to convert these branches now that their code has been incorporated into the event scripts.
10. Review all the remaining functions. If they are very simple, you probably want to embed their code in the event scripts. Uncheck the “function?” box and run step 10 again. Uncheck the “Convert” flag for these.
11. Run the “Replace Functions (View)” query. This will take all the remaining branches which should be marked as functions and replace the branch call with a call to the function. If the results look good, run the Replace Functions (Update) query. There are “1” and “2” versions of this query. Run it multiple times in order to process nested code. When zero rows are updated you are done.
12. Run the “Check for any remaining Branch Calls” query. You’ll need to triage each of these and fix them manually. These can be calls to non-existent branches (comment them out), or bad syntax (fix manually), etc. This could also be the variable branch standard code which can be removed.
13. Create the Script Repository folder structure for this agency, using the sample structure from the master script distribution.
14. We are ready to output these scripts to their individual script files. Run the ExportToTextFiles macro. Give it the path name to the root of your repository folder. If all goes well you should have the scripts populated. Only the “new” functions are created. Next we will get the ones from the custom script areas in AA.



1. Open the existing “INCLUDES\_CUSTOM” script (or events->Custom Script, whichever is used) from your agency and paste the contents into a new file somewhere in your repository (temporarily). We are going to use the FileSplitter macro to separate the functions into individual files. Start the macro, and give it the path to the INCLUDES\_CUSTOM file, as well as the INCLUDES\_CUSTOM target folder. WARNING:
   * This tool struggles with custom JavaScript classes, so move them out into their own files individually first.
   * Any comments with both the word “function” and an open paren “(“ will cause problems. Be sure to check for compilation errors when the INCLUDES\_CUSTOM file is deployed.



1. Manually review each script file and refactor, looking specifically for:
   * Unnecessary global variables
   * Bad coding practices. (not nesting ifs, etc.)
   * Use a JavaScript formatting tool to clean up indentations.
   * Conversion issues
2. Go through the master scripts used for each agency and tease out any custom functions that are used. Save to your INCLUDES\_CUSTOM area.
3. Go through each master script and use a compare tool to see if any changes to standard functions were made. These will also need to be included in the INCLUDES\_CUSTOM folder.
4. Upgrade the agency environment to the latest version of the master script 3.0 distribution from Accela Community. Be sure to make a screenshot of the existing script/event mappings so you can rollback the upgrade.
5. If the agency doesn’t have a repository, you can create one for them using your Assembla account. Connect the agency environment to the repository by setting the EMSEToolConfig standard choice.
6. Make sure the EMSE\_Execute\_Options is set to run scripts only. If both options are checked, the master script will execute both standard choices and scripts.
7. Commit your changes to the repository, use the EMSETool to push the new scripts to your agency and start testing!