Scripting Guide 2023

Intro

‘Scripting’ is just the process of using Photoshop to automate the creation of print and listing images for products. Instead of manually and repetitively mocking up designs on products and creating the print files for each product for a design, Photoshop does it.

Scripting used to be accomplished via a large batch of Photoshop actions. These were a mess to dissect and understand what was happening, and it was difficult to add new products that we decided to sell or remove old products that we were discontinuing. Photoshop actions also made it so that the old ‘script’ had to be run from a PC desktop for a specific username, you had to set the foreground color of Photoshop to your desired background color, and lots of other little, small details that were not obvious and made it really intimidating to create product files and images.

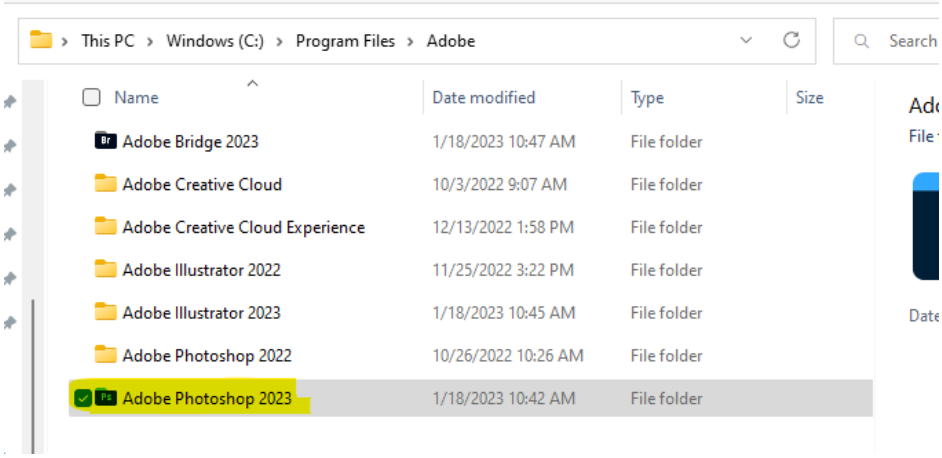
The new version of the script is an actual script that is using JavaScript to tell Photoshop what actions to perform. Because this is now being accomplished with code, logic and conditionals can be added to allow for more customization of what a person may want the script to run. It has a user interface that makes it much easier to a product set up and the ability to choose what products to create.

Listing is a separate action and step from scripting. Listing products (for example on Amazon) is done manually or can be done via spreadsheet on some marketplaces.

Towards the end of this document is some information on how to alter the script. This needs to be done with caution.

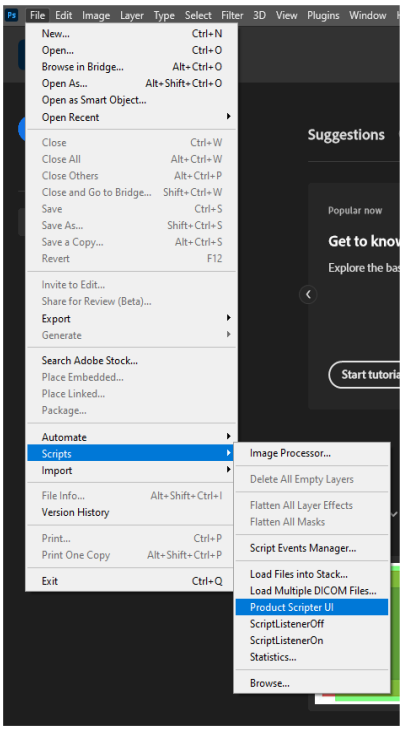
Downloading and Setting Up the Script:

1. Download the Scripting folder.
2. Move Scripting folder to C:/.
3. Locate “Product Scripter UI” file inside of the Scripting folder.
4. Copy and paste (specifically make a copy) “Product Scripter UI” to the computer’s Photoshop Scripts folder.
    a. C:\Program Files\Adobe\Adobe Photoshop 2023\Presets\Scripts
    b. Make sure to select the most recent version of Photoshop that you are using’s folder.



This laptop has both 2022 and 2023 versions of Photoshop. Use the most recent one or whichever version that you use to regularly design with.

5. If you had Photoshop open, restart it so that it can read that it now has Product Scripter UI.
6. You can now access the script in Photoshop via File > Scripts > Product Scripter UI.

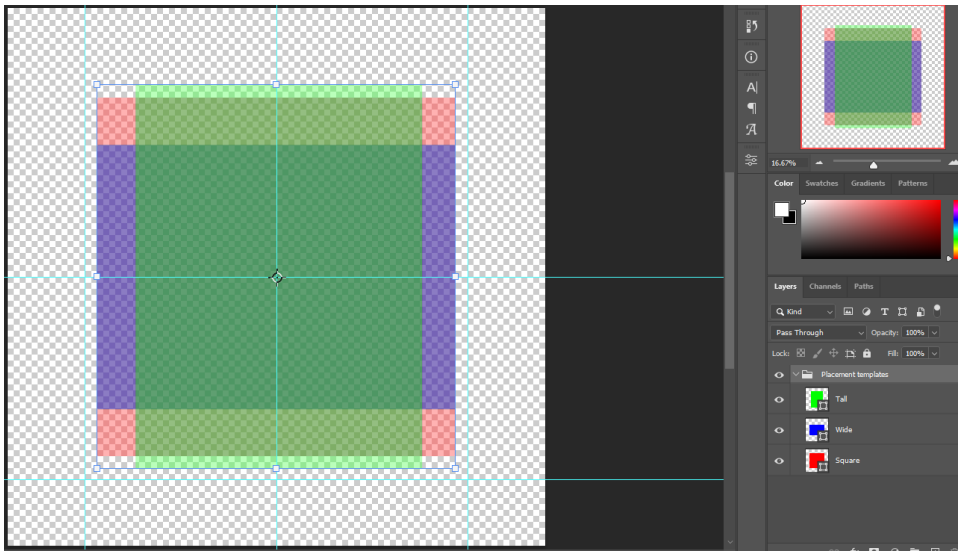


You can access the script and run it through Photoshops menus.

Note: You cannot rename or move the Scripting folder as the file location is hard coded in multiple places.

Preparing a File to go Through the Script

1. Create and finalize your artwork. We will need 3 versions of it to go through the script. Layer names ARE case sensitive.
    a. Have a flattened layer copy of your lines and color without a background, name this 'Color'.
    b. Have a layer that is just a copy of your lines without any background or fill, name this 'EP'.
    c. Have a layer that is the inverted version of your lines, name this 'EP Inverted'. Note that Inverted has a capital at the beginning.
        i. To make this layer, make a copy of your lines that are black with white filling areas that would be colored.
            1. I do this by copying my lines layer, then making a selection from my color layer and filling a brand-new layer with white and merging the lines and fill layers.
        ii. Use an Invert adjustment layer on your lines with white fill and merge.
        iii. Add a black Stroke to the layer (try 15-20 pt) and Rasterize Layer Style.
        iv. Use the Magic Wand with Contiguous unchecked to select and delete the white so that it is transparent.
        v. Note: You can alter the result of your inverted layer and/or make your own, but I would recommend understanding how the result turns out in the first place before tinkering too much. Some designs such as the sports typography ones will have the same for the EP and EP Inverted layers.
    d. Save this file.
2. Open the Scripting folder.
3. Go into the Setup folder.
4. Open the Targeting file in Photoshop.
5. You will see three different colored layers called Tall, Square, and Wide.

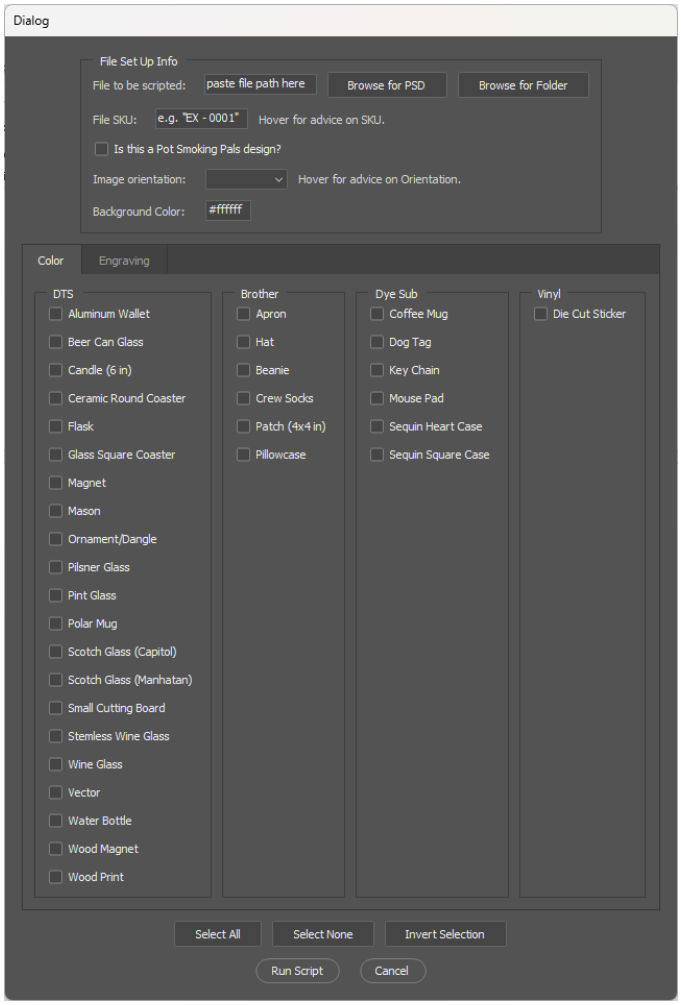


Tall is green, Wide is blue, Square is red.

6. Paste in your 3 versions of your file into this document.
7. With your file's Color, EP, and EP Inverted layers on top of each other, move and resize them so that they fit inside the orientation that fits them best. Give them a small amount of breathing room from the edges. Remember which orientation you use because we will need to put it into the user interface of the script.
8. Delete the folder with the placement templates. Do not resize or crop this document. You want all 3 layers to be existing in the top hierarchy of your document's layers.
9. Double check your layer names. You want Color, EP, and EP Inverted. The order of the layers doesn't matter. Set them all to visible.
10. Save this file somewhere that you can remember.

Using the Script:

1. Open the user interface for the script by opening Photoshop and going File > Scripts > Product Scripter UI. This will open a window in Photoshop (jump to detailed explanation of window).



2. Fill out the information in the window (set your File to be scripted to the file we just made in the last section) and select what products you want to create. It will give you an error if something is not filled out properly.
    a. Don't forget the Engraving tab!
3. Hit Run Script. Photoshop will be performing the actions, but you should be able to use your computer for other things during this time.
4. When the script is done an alert will show up in Photoshop saying that it's done.
    a. If you selected products that need human input, they will be opened and the alert will ask you to adjust them.
5. Files will be in the Outputs folder in Scripting.

Instructions for Files That Need Human Input/Changes:

Most of these will open automatically after the script runs. The Sticker doesn't because it needs Illustrator and the CutContour swatch which not all PCs have and because Illustrator needs different code.

Aluminum Wallet Listing Files

1. Hide/show layers to select what wallet base color will be used.
2. Save a JPG.

Engraved Dog Tags and Key Chains Listing and Print Files

1. Select whether the EP or EP Inverted will be used for the item, delete the layer for the other unused one.
2. Save a JPG for the listing files.
3. Save the PSD for the print files.

Die Cut Sticker Print Files

1. Open the Print file PSD in Illustrator.
    a. If it asks, you can flatten layers to a single image.
2. Open the Print file EPS in Illustrator.
3. Select the path in the Print file EPS (this is your cut line) and copy it.
4. Paste the cut line path in place in your Print file PSD in Illustrator via Edit > Paste in Place or Ctrl + Shift + V.
5. Change the stroke to the cut line to the CutContour swatch if able, and make sure the path is at the top of the hierarchy.
6. Save the file as an EPS.

Ornament/Dangle Listing and Print Files

1. In the print and listing file, position the Color and Fill layers so that they align with the AttachmentCircle layer where you want. You may need to erase some of the Fill layer if it overlaps the hole of AttachmentCircle too much.
2. Save the Ornament Print file as is (a PSD), it will get arranged and set up when needed.
3. Follow the rest of the instructions on the Listing file (you've already done Step 1).
    a. Merge Fill and AttachmentCircle layers and rasterize the layer and any effects on it.

b. Set the merged layer of Fill and AttachmentCircle layer Opacity to somewhere between 50-90, whatever looks realistic with the color but shows some of the wood grain. Make it look like our real product.
c. Select the pixels of the merged Fill/AttachmentCircle layer (Ctrl + click the layer thumbnail).
d. Unlock the Wood layer.
e. Create a mask on the Wood layer from the selection of the merged layer.
f. Hide/show the attachment layers OrnamentString and DangleString as needed and hide the instructions layer. Save each as JPG files for Ornament and Dangle.

Vector Listing Files

1. For both DCS and EP Vector listing files you will need to hide/show what color lighter base you want to have.
2. EP Vector listing files you need to choose and show the engraving type that matches the folder your lighter base belongs to (Brass, Silver, or Metal). Hide the others.
3. Save a JPG for the listing files.
4. Note that if you need to make adjustments to design position on the listing file, go adjust the print file as well and save it as the same PSD.

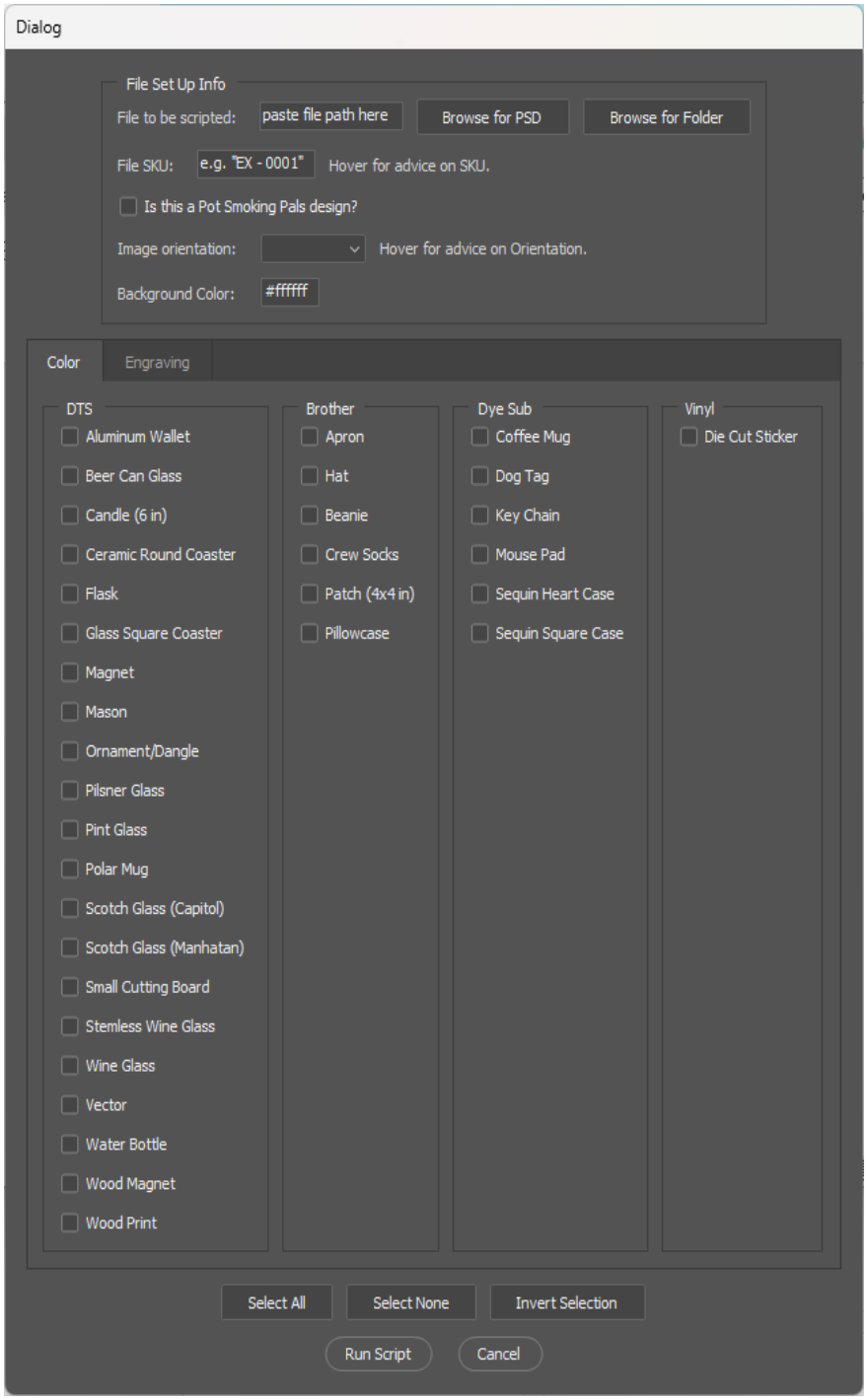
PSP Vector Print and Listing Files

1. For both DCS and EP PSP Vector files, you may need to adjust the position of the logo and/or art in both the print and listing files.
2. EP Vector listing files you need to choose and show which engraving type for both the logo and the artwork that matches the folder that your lighter base belongs to (Brass, Silver, or Metal). Hide the others.
3. Save a JPG for the listing files.
4. If editing the print file, save it as the same PSD.

Water Bottle Listing Files

1. Hide/show layers to select what water bottle base color will be used.
2. Save a JPG.

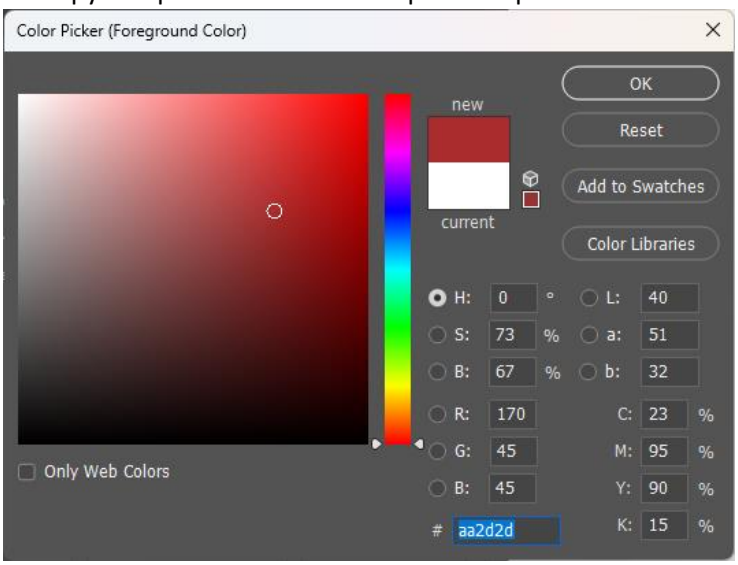
Detailed Explanation of UI



1. **File to be scripted** – This is where you set what file you want to have processed by the script. You can input a single set up PSD file, or select a folder that has multiple set up PSD files in it.
    a. Manually input a file path into the text field via typing or copy/pasting a file location.
    b. Browse through the computer for a PSD file to select.

    c. Browse through the computer for a folder to select.
2. **File SKU** – Input the SKU that you want appended to each file name. Whatever is in this field will go into the file name so do not include the quotes of the example, do not include characters that are invalid for file names, and do not put a space at the end. Examples include NBA - 0024, RO - 0016, or DM - 0022.
    a. When running a PSD file this field is required.
    b. When running a folder this field can be filled out and use the SKU as a prefix for the variation names inside of the folder.
        A. Example: I have a folder for the design DM - 0022 that has variations and input DM - 0022 in the File SKU field. Each variation is inside of the folder in its own PSD and is named the color of the variation such as RED, GREEN, BLUE. The output files for an EP Pint will be EP - PINT - DM - 0022 - RED, EP - PINT - DM - 0022 - GREEN, EP - PINT - DM - 0022 - BLUE.
    c. When running a folder of unrelated files (not a variation) you can leave this field blank and it will use only the names of the files.
        A. Example: I have a folder of new NFL designs. Each file is inside of the folder as its own PSD named NFL - 0040, NFL - 0041, and NFL - 0042. Since I didn't input a File SKU, when the script processes the files inside of this folder it will use the names of the files. The output files for an EP Pint will be EP - PINT - NFL - 0040, EP - PINT - NFL - 0041, and EP - PINT - NFL - 0042.
3. **Is this a Pot Smoking Pals design?** – Checkbox for if this is a PSP design. If it is, it will make changes for certain items that are different for PSP designs. This includes setting a tiled logo background or adding the logo to the print and/or listing file.
    a. Items that have PSP changes:
        A. DS ceramic mugs
        B. DS mousepads
        C. BR 4x4 in patch
        D. DTS rectangle magnets
        E. DTS ceramic (round) coasters
        F. DTS wood print
        G. DTS Vector
        H. EP Vector
4. **Image orientation** – Set the orientation for the artwork. Remember when we matched the artwork to one of the colored rectangles when setting it up? The orientation that you picked is what you'll select here.
    a. When we run a folder, we can only pick one orientation to apply for the whole folder. Variations are likely similar in orientation so that isn't usually a problem, but if running unrelated files I would recommend sorting and running them by orientation. Instead, you can put all of your Tall files in one folder separate from your Square files and run your Tall folder with the orientation set to Tall, then run your Square folder with the orientation set to Square.
5. **Background Color** – Enter a hex code for the color you want as the background for certain items. This field is ignored for PSP files. It accepts a hex code with or without a #, but it does require a 6-digit hex code and will not accept 3-digit shorthand formats.

a. Tip: You can just copy and paste from Photoshop's color picker.



6. **Product Tabs** – Select what products you want to produce for a file. There are tabs for each production type of Color and Engraving, and then each tab is separated into departments if there are multiple departments for a production type.
7. **Selection Buttons** – For ease of use, selection tool buttons have been implemented.
8. **Run/Cancel Buttons** – Buttons to start the script or cancel and get out of the Product Scripter UI.

Changing the Script

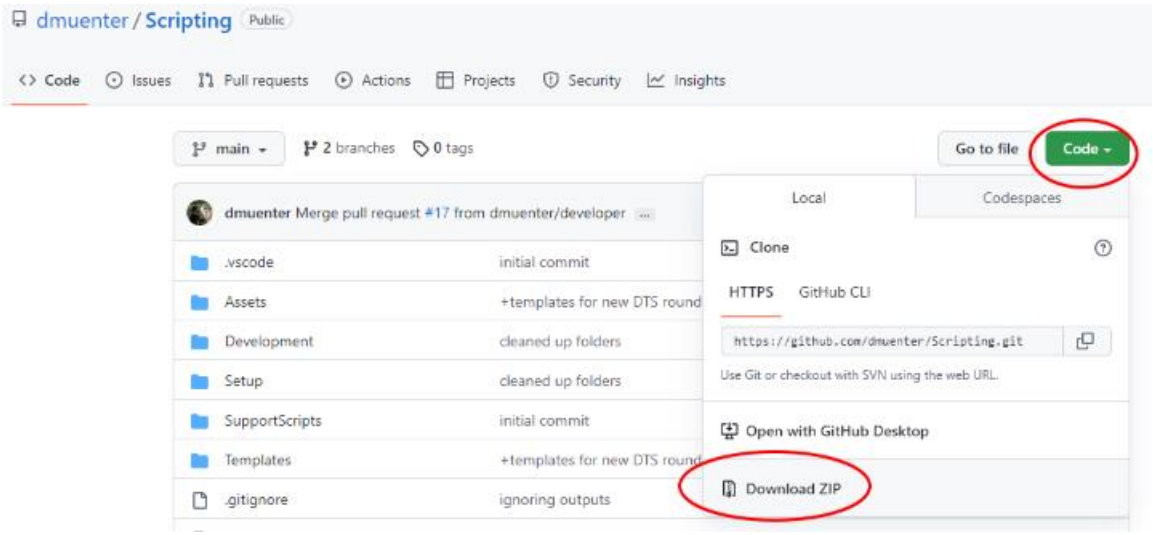
Mega Disclaimer and Information

Since the script is just code, it is possible to change the script to add or remove products. This needs to be done VERY CAREFULLY! Create backups, make save iterations to know where things went wrong if it isn't working, etc.

The script was written using the JavaScript language in an application called Visual Studio Code or VS Code. Inside of the Scripting folder there is a Development folder that has some notes and resources. The information on how to change the script is going to be a lot less categorized because a lot of it happens in tandem.

This script was written by someone who took and struggled through two coding classes in college (neither were JavaScript) and who has a professional software engineer in their pocket. I worked on the script from August 2022 to February 2023 with a lot of help…this is difficult material.

If something happens that rewrites the script files in a bad way…redownload it from Google Drive, and if that version is bad, the last version I updated will be in GitHub.



How the Script Works, short summary

When running the script through Photoshop, it has a user interface (UI) that collects information that you put in. When the script runs, it takes that information and applies it where needed, and runs the functions inside of the script for each product to be built.

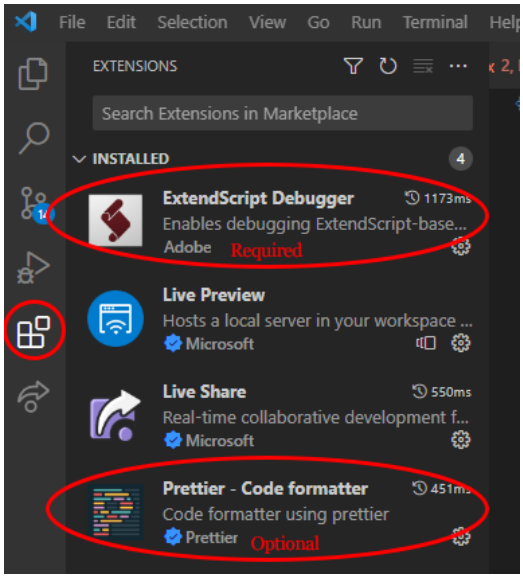
When a product is being built, it has template files that get opened. Each template has a folder of Targets, which are the colored orientation rectangles (Tall, Square, Wide) that match the rectangles that artwork is set up in. When the product is being built, Photoshop will replace the rectangle of the correct orientation with the art file that you plugged in. It then will keep and delete the appropriate layers from Color, EP, and EP Inverted. Each product print and listing template needs different edits to be properly

set up. These steps are done by the script and then the resulting print or listing files are saved in the appropriate folders inside of the Outputs folder where you can find them afterwards.

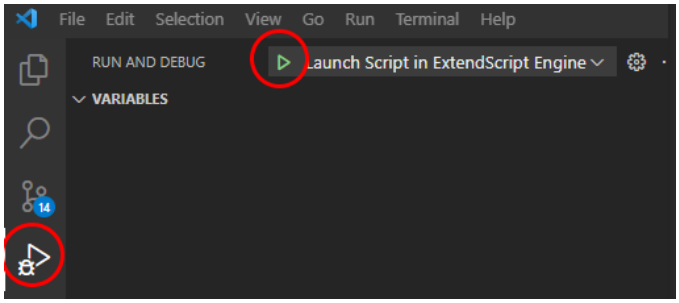
Setting up VS Code

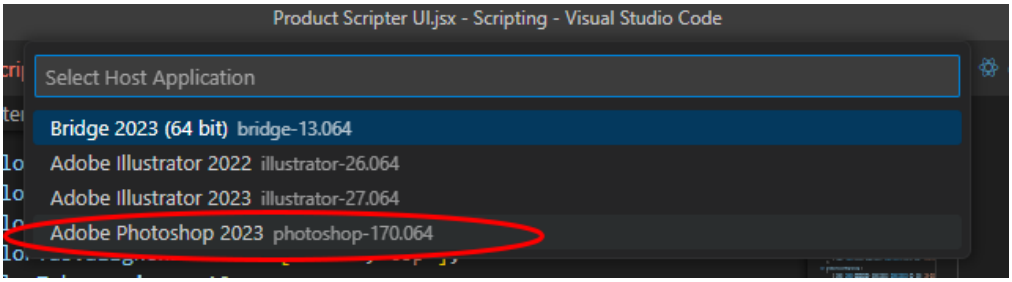
If you want to edit the script, then you will want to try and use a coding application. Since this was built in VS Code, I would say use VS Code which can be installed here: https://code.visualstudio.com/

Once installed and open, there is an extension needed for testing called ExtendScript Debugger. Photoshop's specific brand of JavaScript that it uses is called ExtendScript. Access the Extensions by clicking the icon on the sidebar with the squares, and then search the Extensions in the bar. Click on the result that you want, and there should be an install button to add it to your VS Code.

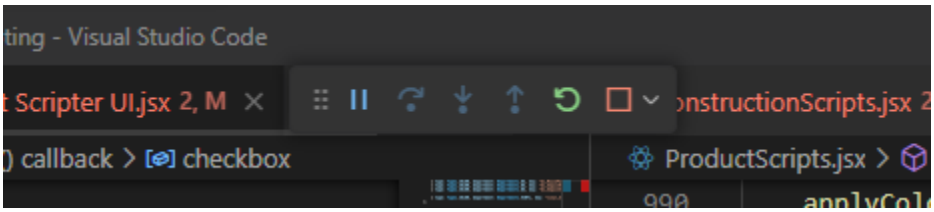


Once this is all set up, we can test the code from VS Code. You need to have your Photoshop open, and in VS Code you will want to have "Product Scripter UI.jsx" open and the most recent tab that you've clicked in the program. In the sidebar, there's the Run and Debug tab (with a play button and a little bug). When you're in that tab, there's a play button near the top. The dropdown needs to be set to Launch Script in ExtendScript Engine. When you click the Play Button, a dropdown will appear in the top middle of the screen. Select your Photoshop version that's open.





This should run your UI in Photoshop if you switch to Photoshop. If your script is running, there should be a little bar that has a pause or stop option in the top of your VS Code window, as well as an orange bar across the bottom of the VS Code window.



If you need your test to stop, just press the Square to stop it in the bar at the top and it will stop running the script.

If you add a product in the order that the guide is in, the first thing that you'll test is seeing if your product gets added to the UI. After that, you'll want to try testing once you've gotten to the second part of editing in ProductScripts.jsx when you write your function. Try testing to make sure that the correct templates open when you add your file paths and every step or every other step after that to make sure that it is doing what you wanted.

When we set up the script originally, we kept "Product Scripter UI.jsx" inside of our Scripting folder and made a copy of it that goes in the Photoshop Scripts folder. Because of this, when making edits, it will effect the version that is inside of the Scripting folder. You will test with that copy of the file through VS Code. Once everything is successful, update the UI that runs from Photoshop by copying the file from the Scripting folder to the Photoshop Scripts folder again.

Introducing the Photoshop Templates

There are two main template files for any given item. There is a print template, and a listing template. Both templates will have a Targets folder holding the orientation rectangles. These Targets folders will be positioned on the template where the artwork needs to go for both print and listing files.

If an item is being produced on multiple stations (e.g. DTS and EP) then the print template files need station prefixes to separate the EP formatted print template and the DTS formatted print template as the stations do not generally share templates.

Listing templates of an item produced on multiple stations can sometimes be shared, but it depends. DTS and EP Pints likely put the design in the same location on the glass, but DTS and EP small cutting boards have different locations for the art, and thus need different templates to place the target rectangles correctly for the respective item.

If you are adding a new product, you may need to add a new folder inside of the Templates folder for the product, and then create templates for the print and listing files. The goal is that print templates that come out of the script are print ready. For some stations like EP and Brother, they can just hide unwanted layers and send the print as it currently looks so extra layers isn't a huge issue. Other stations like the old DCS printers cannot have hidden layers and need the artwork to be flattened. There is some nuance to setting up these files that will be detailed later, but you can look through the files in the Templates (be careful not to save over them) to get an idea of how they look.

Introducing the Script Files

The script is run in Photoshop using the "Product Scripter UI.jsx" file. This file links to "main.jsx" and "extendscript-es5-shim.jsxinc". The shim is essentially an add-on file that includes some functionality that Photoshop's specific brand of JavaScript doesn't have built in.

main.jsx is the main hub of the program. When you tell the script to run, main.jsx is what says what runs. It links to the shim and a file called "ProductScripts.jsx".

ProductScripts.jsx has two purposes. The first purpose is to map which function runs corresponding to which product checkbox from the UI. The second purpose is to lay out the steps that each product takes to create their files.

ProductScripts.jsx has two included files. One is "Unsmart.jsx" which is a script that swaps out the file for a Smart Object. This script just hangs out and should never need to be altered. It also includes "ConstructionScripts.jsx" which is where all the functions for the steps that make up a product are defined.

If you are adding or removing a product, there are two script files that you need to change. In this script, an engraved item is a separate product (and separate checkbox and function) than a color printed version of the same item.

1. "Product Scripter UI.jsx" needs changes to add/remove a product's checkbox in the UI.
2. ProductScripts.jsx needs changes to add/remove the mapping of which checkbox activates which function and to lay out the steps for creating a product's files.

Creating Photoshop Template Files

The core of making a new template file is placing the orientation rectangles, what I've been calling targets. These targets are the same scale as the orientation rectangles in the Targeting file, that way when we switch them the art will sit exactly where our orientation rectangles did.

Inside of our Setup folder there is a Staging folder. There are three PSDs inside of that folder, orientationSquare, orientationTall, and orientationWide. You will want to place these into your template file, either by dragging all three from the file explorer in or via File > Place Embedded. Grab all three of these and place them into a folder named "Targets" which, like a lot of our other layer or group names, is case sensitive. Once they're in the Targets folder, use the group to move the rectangles into place or scale. If you want or need to scale them individually, you can do so, but you want to be careful to keep them proportional. To keep the same proportion for the other template that you create for the product,

right click on the Targets group and click Duplicate Group, then select the other file in the dropdown that you want to send the copies to. You can continue to move and scale via using the group, but remember that if you change the individual rectangles, it won't match the proportions that you set in your other file.

Depending on the template and the station it belongs to, there may be additional layers to the template that you want to keep. You will want to lock those layers or groups so that they survive a part of the code that will delete everything except for the artwork layer that's chosen and any locked layers or groups. Sometimes the Background layer, even though it has a lock on it, can be deleted so try making it into a layer and then locking it again if you're encountering issues.

I'd recommend looking at how current templates are set up and following the lines of their existing code to see if you can figure out how it interacts with the other layers and groups that may be in the templates.

Adding a New Product – "Product Scripter UI.jsx"

What needs to be done in "Product Scripter UI.jsx" when adding a product is adding the product into the Photoshop UI so that it can be seen, interacted with, and send information to the rest of the script when it is used in the UI so that the rest of the script knows what to do to build that product.

Near the top of "Product Scripter UI.jsx" there are two lists, one for *productsListColor* and one for *productsListEngraving*. Each list is an array (a type of data that holds a list of things, separated by a comma) of objects, and each object (the 'thing' that we are holding in our array) has the information for each product that tells it where to display in the Photoshop UI. Each Department's products are manually listed in roughly alphabetical order or related products are next to each other.

Color printed products include color UV LED printing, Brother, dye sublimation, and vinyl stations. Engraving products are the EP department. The departments for color are separated into blocks visually to keep things easy to read.

Add a new product by adding a new line in the appropriate list and department. The type and department properties will be the same as other product objects so that can be copied. Note that properties are all separated by commas.

```
{ type: Types.COLOR, department: Departments.DTS, key: "dts_AluminumWalletCheckbox", label: "Aluminum Wallet" },
```

For example: if adding a new DTS product, type and department will be the same as other DTS products.

a. type: Types.COLOR
b. department: Departments.DTS

The things that do change for each product is the key and label. The key is the name of the checkbox that our next script will read to identify the product. Our format for the key is a lowercase abbreviation of the department followed by an underscore, the name of our product, and "Checkbox". This needs to be unique to the product and is case sensitive. The label is what displays next to the checkbox in the UI. The label does not have to be unique, for example, if you are adding both a color and engraved version, because it can be identified by the production type and department.

So, if I were to add a new DTS product that is a wooden keychain:

a. key: “dts_WoodKeyChainCheckbox”
b. label: “Wooden Keychain”

The double quotes are needed to denote that this information is a string.

So, the whole line of our added wooden keychain product should be:

```
{ type: Types.COLOR, department: Departments.DTS, key: “dts_WoodKeyChainCheckbox”, label: “Wooden Keychain” },
```

Make sure to pay attention to the commas in between each property, and make sure that you include the comma after the ending curly brace.

Adding this line for the new product is all that needs to be done in “Product Scripter UI.jsx”. If you wanted to test it, you can run the UI from Photoshop and see if your checkbox has appeared in the correct place. If you made a copy of the UI to alter, you will need to move the copy into the Scripts folder like when the script was set up on the PC. You can name the file differently like “Product Scripter UI update test” and run that one through Photoshop.

Adding a New Product - ProductScripts.jsx – Part 1, adding the part that reads the UI

Two things need to happen in ProductScripts.jsx when adding a product. The first thing is that we need to add to our array of *ProductScripts* the new item that we added to our UI and what function (a named ‘group’ that contains a set of statements that performs a task) occurs if it is selected and run. The second thing is writing the function for the product, which includes providing information on what templates to use and the steps needed to create the print and listing files.

At the top of ProductScripts.jsx, there is a defined constant (a piece of data that we create with code where the values assigned to it cannot be changed by the program) called *ProductScripts*. This constant is an array of objects with each object holding the key and the *functionToRun* for a product. The key we created in the last step when we added our product to the UI. The *functionToRun* is a property that tells us what function goes with that product key.

```
{
    key: "dts_AluminumWalletCheckbox",
    functionToRun: process_DTS_AluminumWallet
},
```

We need to add in our product that we just created in the UI to match. The products in this list are not particularly organized so you can add it at the end or right next to a product from the same department where you can copy from an example.

Two things need to be added, the key that we just made, and then we need to create a name for the *functionToRun*. The *functionToRun* name format that we’ll use is “process” followed by an underscore, capitalized abbreviation of the department, underscore, and then a name referring to the product. You can use what was used in the key without the “Checkbox” part if that’s easiest.

To continue our DTS wooden keychain example:

a. key: “dts_WoodKeyChainCheckbox”
b. functionToRun: process_DTS_WoodKeyChain

Our example should look like this:

```
{
     key: "dts_WoodKeyChainCheckbox",
     functionToRun: process_DTS_WoodKeyChain
 },
```

Like adding an object in the array in the last step, make sure that you separated each property with a comma and that you remember the comma after the closing curly brace.

Adding a New Product - ProductScripts.jsx – Part 2, adding the instructions to make the product

Next, we need to create the function that we just gave a name to and write what tasks it performs. This part is trickier. This section is going to have more examples of the existing code and how to interpret it instead of making a parallel of our DTS wooden key chain example every time.

When writing the existing functions, I broke down how I would perform the task manually in Photoshop into steps. Some steps are grouped together to be performed all at once (e.g. tedious or repetitive steps that can be used across multiple or all products), some steps are individual. It depends on what needs to be done for the product. Try to think about if creating the files for a new product would match the steps as an existing product, that way you can just mimic an existing product vs having to figure out the steps yourself and stitch things together.

Here is an example function for the DTS Aluminum Wallet:

```
function process_DTS_AluminumWallet(data){
    var printTemplatePath = "C:/Scripting/Templates/AluminumWallet/Aluminum Wallet - Template.psd";
    var listingTemplatePath = "C:/Scripting/Templates/AluminumWallet/Aluminum Wallet - Listing Template.psd";
    var savePrintDestination = "C:/Scripting/Outputs/" + data.fileSKU + "/Print Files/DTS - AW - " + data.fileSKU;
    var saveListingDestination = "C:/Scripting/Outputs/" + data.fileSKU + "/Listing Files/DTS - AluminumWallet  - " + data.fileSKU;

//Generate Print File
    openTemplate(File(printTemplatePath));
    processFile(data);
    keepColor("print");
    deleteAllFolders();
    savePSD(activeDocument, File(savePrintDestination));
    closeDocument();

// Generate Listing File
    openTemplate(File(listingTemplatePath));
    processFile(data);
    keepColor("listing");
    unlockAllFoldersExcept(["Bases"]);
    deleteAllFolders();
    if (data.imageOrientation === "Wide") {
        activeDocument.rotateCanvas(-90);
    }
    savePSD(activeDocument, File(saveListingDestination), true);
    closeDocument();
}
```

The majority of ProductScripts.jsx is functions for each product. They are broken up into 3 main sections visually, kind of like paragraphs:

a. File paths
b. Print file creation
c. Listing file creation

To create a function, you declare it by saying “function” followed by the name of our function that we just created for *functionToRun*. Add a set of parentheses to establish your parameters, which is data or information that the function can take in and use. All of our ProductScripts functions will take a variable (a piece of data that can have its contents changed by the program) named *data* that is created and given information that we filled out in the UI when run. Lastly, you will open and close the contents of your function with curly braces ({}). All of the information and steps that belong to our product will now go inside of our function, meaning between those curly braces.

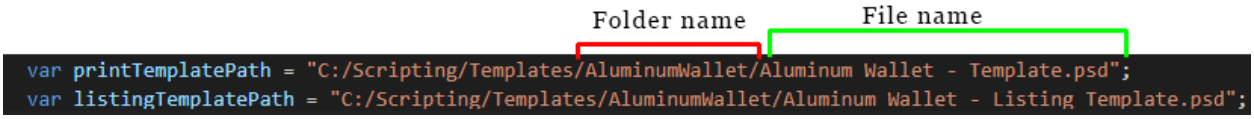
File Paths

If you’ve made your Photoshop templates already, then you have the exact file locations to insert into the file paths section. If you haven’t made them yet, you can write in what you will name them, or add them in after you create the templates. Since these file locations are strings, they are case sensitive and must be exact.

```
var printTemplatePath = "C:/Scripting/Templates/AluminumWallet/Aluminum Wallet - Template.psd";
var listingTemplatePath = "C:/Scripting/Templates/AluminumWallet/Aluminum Wallet - Listing Template.psd";
var savePrintDestination = "C:/Scripting/Outputs/" + data.fileSKU + "/Print Files/DTS - AW - " + data.fileSKU;
var saveListingDestination = "C:/Scripting/Outputs/" + data.fileSKU + "/Listing Files/DTS - AluminumWallet  - " + data.fileSKU;
```

There are four file paths that you will need to create and write out. Two are for the print and listing templates, and then two are the save destinations for the print and listing output files.

You will need to declare the variables for the *printTemplatePath* and *listingTemplatePath* and give them their values, a string of the file location for each template. Since this is a string, it needs to be enclosed in double quotes. My rule for creating and naming the templates is that the print template is just “Template”, and the listing template is “Listing Template”. These are the files that the script will open for this product and make changes to so that it can create the print and listing files. Make sure that you include the name of the folder that the files are in.



Notice that in our save destination file paths, we’ve inserted + signs and *data.fileSKU*. Using the + signs with strings of text and variables is known as concatenating, or basically just adding stuff together into a string. Because the save folder in Outputs depends on what we put into the UI when we ran the script, we have to use the variable data for the SKU when constructing our file path.

```
var savePrintDestination = "C:/Scripting/Outputs/" + data.fileSKU + "/Print Files/DTS - AW - " + data.fileSKU;
var saveListingDestination = "C:/Scripting/Outputs/" + data.fileSKU + "/Listing Files/DTS - AluminumWallet  - " + data.fileSKU;
```

So for a SKU of DM - 0022, this will make the file path for our *savePrintDestination* "C:/Scripting/Outputs/DM - 0022/Print Files/DTS - AW - DM - 0022. Basically it will insert whatever we put into the SKU field wherever it says *data.fileSKU*.

Our save destinations will be creating brand new files, so you don’t have to exactly match an existing file name. I do differentiate between the print file and the listing file by using the actual product SKU format

for the print file (all caps, may be abbreviated, etc) and then writing out the name of the item in the listing file. This is because when designers are looking for a file for a product, they will have the SKU for the product on the ticket, so it's easier to read and understand what they need to look for.

Print File

Creating a print file can be broken up into some very general strokes.

1. Open my print template file.
2. Swap out the orientation placeholders with my actual artwork.
3. Delete any extra layers such as unneeded orientation placeholders.
4. Perform any edits that need to happen to the file.
    a. This could be applying #404040 gray to a glass file, or creating duplicates of my artwork for a template that holds multiple sets such as a mug file.
5. Save my new print file.
6. Close the file.

Depending on what production method and how the template is set up, the print file creation can be pretty short and sweet (most average color files), have a few edits (most EP files for glassware), or have a lot of steps (DS ceramic mug file). All of the initial placement of the artwork is achieved by the orientation rectangles and switching the artwork out. The rest can vary. Each of those general strokes is labeled on the following existing products.

In the screenshots below, the above general steps have been labeled on the code lines of existing products in the script.

DTS Aluminum Wallet

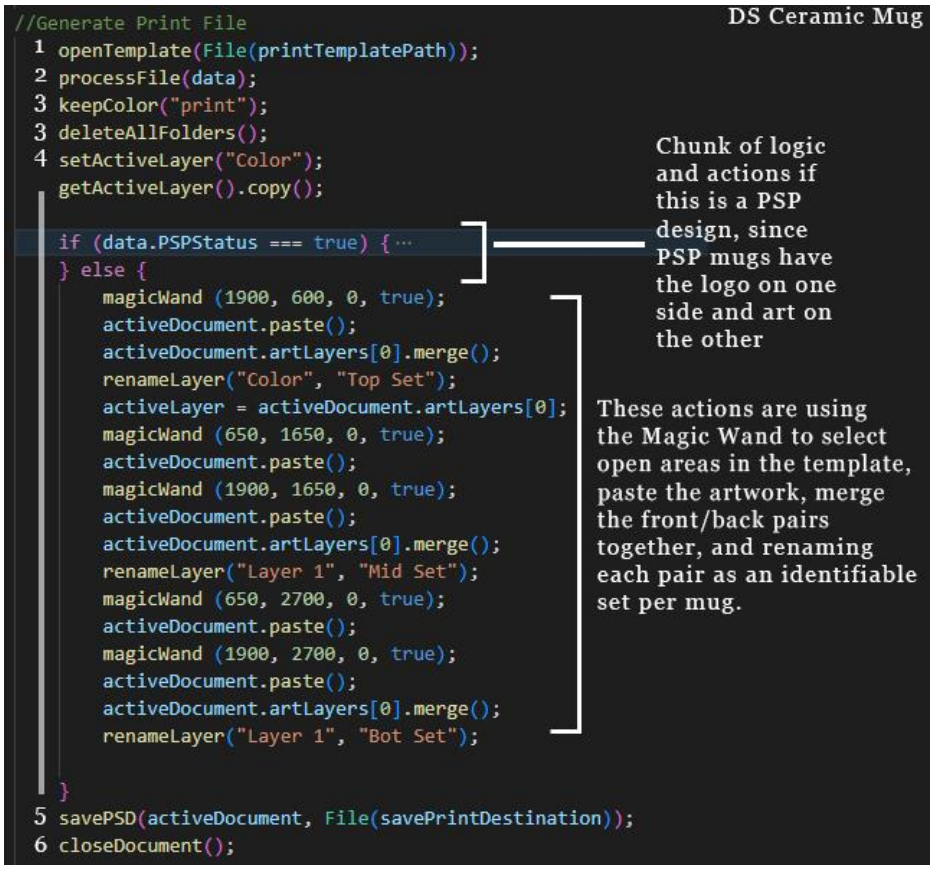
```
//Generate Print File
1  openTemplate(File(printTemplatePath));
2  processFile(data);
3  keepColor("print");
3  deleteAllFolders();
5  savePSD(activeDocument, File(savePrintDestination));
6  closeDocument();
```

The DTS Aluminum Wallet print file instructions are very basic. There is no extra editing that needs to be done other than placing our artwork and removing any unnecessary layers.

EP Pint Glass

```
//Generate Print File
1  openTemplate(File(printTemplatePath));
2  processFile(data);
3  keepEPInverted("print");
3  deleteAllFolders();
4  apply404040("EP Inverted");
5  savePSD(activeDocument, File(savePrintDestination));
6  closeDocument();
```

An EP Pint Glass print file is similarly simple, but since glassware needs the #404040 gray, we need to apply that to our EP Inverted design layer.



The DS Mug, while similar to the DTS Aluminum Wallet where it just needs the Color layer, has a lot of editing steps. It needs more than one copy of the artwork which we have to paste the artwork into specific areas of the template (by telling the Magic Wand tool which exact coordinates to click) to set up.

Listing File

The listing file creation is approached in a similar manner to the print file, but depending on the product and template will need different editing steps.

Reiterating the general strokes with some listing context edits:

1. Open my listing template file.
2. Swap out the orientation placeholders with my actual artwork.
3. Delete any extra layers such as unneeded orientation placeholders.
4. Perform any edits that need to happen to the file.
    a. EP items sometimes need editing and effects to look like they are engraved onto the product photo, some items have product photos on different colors, depending on the orientation of the picture the product photo will be rotated, etc.
5. Save my new listing file(s).
6. Close the file.

Some examples from existing products are below with the steps labeled again.

BR Patch

```
//Generate Listing File
openTemplate(File(listingTemplatePath));
processFile(data, getActiveLayer());
keepColor("listing");
deleteAllFolders();
layerLock("Fill", false);

if (data.PSPStatus === true) {
    setPSPBackground(data);
    activeDocument.artLayers[0].merge();

} else {
    var bgRGB = convertHextoRGB(data.backgroundColor);
    applyColorOverlay("Fill", bgRGB);
    setActiveLayer("Fill");
    rasterizeLayer();
    if (bgRGB.r <= 35 && bgRGB.g <= 35 && bgRGB.b <= 35) {
        setFillOpacity("Fill", 90);
    }
    selectLayerPixels("Color");
    activeDocument.selection.invert();
    activeDocument.selection.expand(1);
    makeMask("Fill");
    getLayerByName("Fill").blendMode = BlendMode.MULTIPLY;
    getLayerByName("Color").blendMode = BlendMode.MULTIPLY;
}
saveJPG(activeDocument, File(saveListingDestination));
closeDocument();
```

Actions for if it's a PSP design

Actions for if it is not a PSP design

The BR Patch is an item that has differences if it is PSP or not. At the beginning of the Patch function there's some logic that if it's PSP it will switch our template files to open to PSP specific templates. Once it is creating the files, there's an if-else statement that says what actions are to be performed if it's a PSP design (make the PSP background visible and merge it with the artwork), otherwise or else we will perform the steps for a normal design. For a normal design, we apply the *backgroundColor* that we set in the UI to the layer called Fill, then there's another if statement that automatically lightens the fill if it's too dark (just for visual tweaks), then it creates space in the Fill layer that fits our artwork so that the artwork when we apply the Multiply blending mode doesn't show the color of the fill through it. Then we set both the Fill and Color artwork layers to Multiply.

BR Pillowcase

```
//Generate Listing File
openTemplate(File(listingTemplatePath));
processFile(data);
keepColor("listing");
deleteAllFolders();
saveJPG(activeDocument, File(saveListingDestination));
closeDocument();
}
```

BR Pillowcase is very simple comparatively, as we just need to show our Color layer and remove all of the unimportant stuff.

```
//Generate Listing File                                    EP Key Chain
1 openTemplate(File(listingTemplatePath));
2 processFile(data);
3 keepBothEP("listing");
3 deleteAllFolders();
4 MoveLayerTo(getLayerByName("EP"), 1967, 2962);
  MoveLayerTo(getLayerByName("EP Inverted"), 1967, 2962);
  setLayerVisibility(activeDocument.layerSets.getByName("Dog Tag"), false);
  setLayerVisibility(activeDocument.layerSets.getByName("Key Chain"), true);
  applyColorOverlay("EP", convertHextoRGB("#c8c8c8"));
  applyColorOverlay("EP Inverted", convertHextoRGB("#c8c8c8"));
5 savePSD(activeDocument, File(saveListingDestination), true);
6 closeDocument();
```

EP Key Chain has some nuance as the key chain and dog tags share a listing file. The key chain artwork needs to be in a different location than the dog tag however, and the pre-placed rectangles are on the dog tag. This listing file has steps to move both the EP and EP Inverted layers that we kept to the location of the key chain. Then we hide the dog tag folder, show the key chain folder, and apply a color to the EP and EP Inverted art so that it looks engraved.

We kept both the EP and EP Inverted so that we can choose which design will work better for a dog tag afterwards. When the script runs products that need human assistance at the end, it opens up the files that need changes. To mark that a PSD file that we save needs human judgment, we add the “true” at the end of the savePSD line (step 5). Most of our other screenshot examples do not have this and so they do not get added to the list of files that need to be opened at the end.

If you’ve been going in order, once you finish plugging in the steps to create the listing file for your new product, you’re done. You can try running the UI and inputting information, then selecting your new product to test if it works or not.

General Notes to Think About When Adding Products

Here are some things to think about or keep in mind when adding the steps in to create a product.

Tall/Square orientation product rotation vs Wide orientation product rotation

Does your product have a different rotation depending on if you have a Wide orientation vs a Tall or Square orientation? Examples include DTS rectangle magnets or EP small cutting boards. Wide orientations will usually have the product so that the product is horizontal and the art goes along the longest side. Tall/Square orientations usually go with a vertical product so that the longest side fits the artwork that way. I would recommend defaulting your product’s rotation to fit the vertical alignment, and then if and only if it’s a Wide orientation artwork, rotate the canvas. There’s an if statement in the example products’ listing file code that accomplishes this. You’ll want to use this after most to all of the editing has been done, but before saving.

```
if (data.imageOrientation === "Wide") {
    activeDocument.rotateCanvas(-90);
}
```

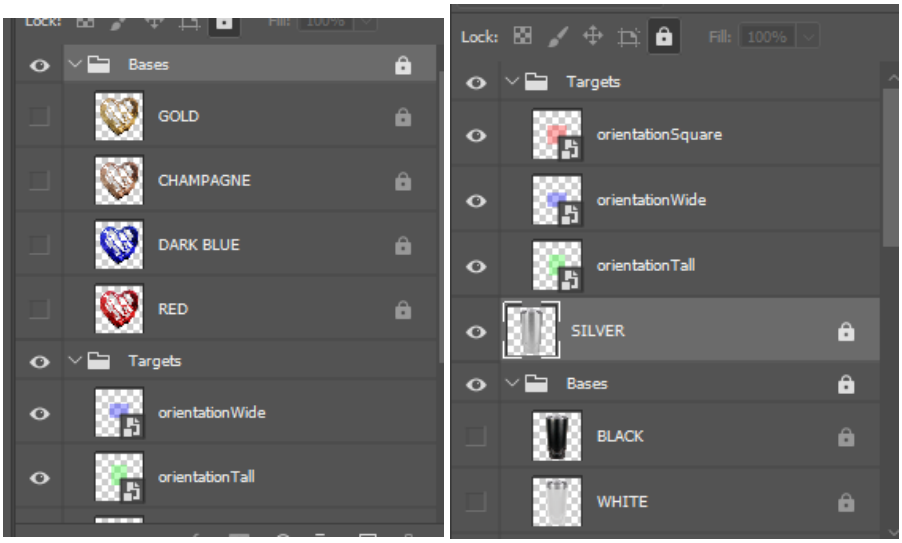
Multiple variations of listing images

Some items we have listing images that are different colored bases and we want to have a copy of all of them instead of manually pick a one. This is true of the Polar Mugs and of the Sequin Heart pillowcases,

for example. Some template set up will need to be done, but there is a function called *processBases()* that will go through the contents of a folder in your file called “Bases” and save a file for each layer inside. You’ll want to do your preparation before using *processBases()* such as keeping the artwork layer that you want or any editing.

```
getLayerByName("Color").blendMode = BlendMode.MULTIPLY;
var saveLocation = saveListingDestination + " - UNCOVERED";
saveJPG(activeDocument, File(saveLocation));
processBases(saveListingDestination); //Don't need File() as it's inside of the function
```

This is a snippet from the DS Sequin Heart pillowcase. Line 1 is doing some editing to our artwork layer that we kept. The Sequin Heart also needs to save a copy of the listing file where the artwork is applied to our white side of the pillowcase and is completely uncovered, so we create a file path name by adding “ - UNCOVERED” and save a JPG to start. Then we can use *processBases()* and we have to make sure to give it our *saveListingDestination.*



Our DS Sequin Heart listing template (left) has a folder named “Bases” with named layers inside. This folder needs to be locked at first so that it survives some deletion that happens when we keep our artwork layer. *processBases()* will methodically go through each layer and append its layer name to the file name when it saves that version. So it will save DS - SEQUIN HEART - SKU - #### - GOLD, DS SEQUIN HEART - SKU - #### - CHAMPAGNE, etc.

EP Polar Mug listing template (right) has SILVER outside of the Bases folder. The code for the listing image manually saves the SILVER version with the EP artwork first. To prepare for the rest of the color variants it hides EP, makes a selection of EP Inverted and applies that selection as a mask on SILVER to create the effect of removing the coating. We have to make SILVER our active layer so that it doesn’t get hidden repeatedly in *processBases()*. Then we call *processBases()* and feed it our save destination and it will hide/show/save for each layer inside.

Pot Smoking Pals

Pot Smoking Pals designs have differences like changing out a background. To set this up, you can make a separate PSP template for print or listing templates. The system I went with for existing products is

that a PSP specific template is just the normal template path, but with " - PSP" added to the end. A function has been built to insert that extra text automatically, but we want to make sure it only happens if we have marked that our artwork will be a PSP design.

```
function process_DTS_Magnet(data) {
    var printTemplatePath = "C:/Scripting/Templates/Magnet/Magnet - Template.psd";
    var listingTemplatePath = "C:/Scripting/Templates/Magnet/Magnet - Listing Template.psd";
    var savePrintDestination = "C:/Scripting/Outputs/" + data.fileSKU + "/Print Files/DTS - MT - " + data.fileSKU;
    var saveListingDestination = "C:/Scripting/Outputs/" + data.fileSKU + "/Listing Files/DTS - Magnet - " + data.fileSKU;

    if (data.PSPStatus === true) {
        printTemplatePath = switchToPSPTemplate(printTemplatePath);
        listingTemplatePath = switchToPSPTemplate(listingTemplatePath);
    }
```

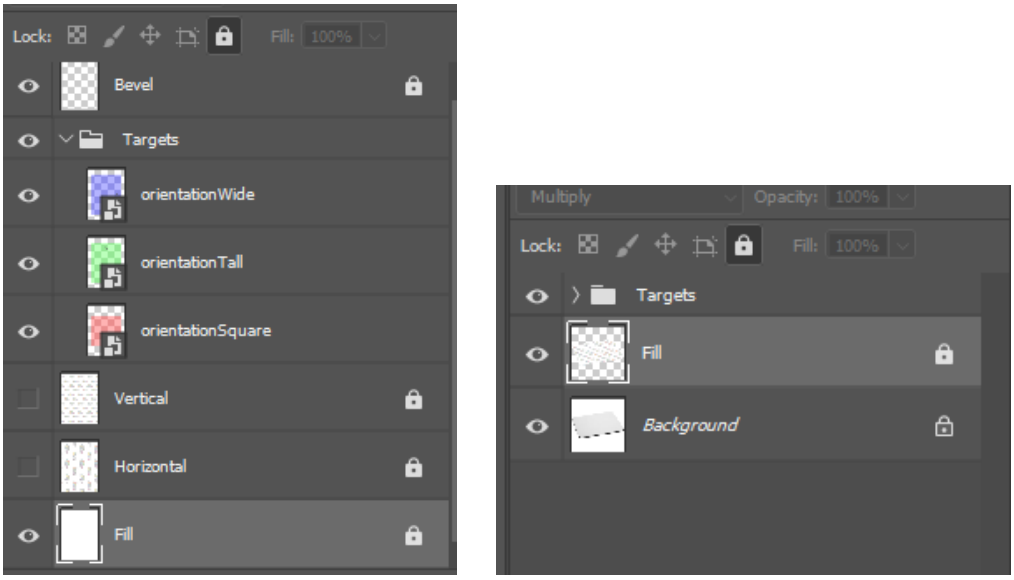
Right after we set our file path variables but before we start the instructions for creating a print file, we need to create an if statement that "if the PSP checkbox in our data is true, we want to switch our template paths to PSP." Most products will need both the print and listing templates to be changed, but if one doesn't need a change (e.g. the ceramic mug since it shows the front only) you can remove the line for that template.

Once we get into the steps to make our print file, we have similar logic in an if-else statement. "If the PSP checkbox in our data is true, we want to make our PSP Background layer visible (this function automatically determines if we need a horizontal or vertical background), else/otherwise if it is not true we will change the color of the fill to match the background color hex code that we set."

```
//Generate Print File
    openTemplate(File(printTemplatePath));
    processFile(data);
    keepColor("print");
    deleteAllFolders();
    layerLock("Fill", false);
    if (data.PSPStatus === true) {
        setPSPBackground(data);
        activeDocument.mergeVisibleLayers();
        layerLock("Color", true);
        unlockAllLayersExcept(["Color"]);
        deleteAllLayersExcept([]);
    } else {
        applyColorOverlay("Fill", convertHextoRGB(data.backgroundColor));
        setActiveLayer("Fill");
        rasterizeLayer();
        activeDocument.artLayers[0].merge();
    }
    savePSD(activeDocument, File(savePrintDestination));
    closeDocument();
```

In the above screenshot for a DTS Magnet, you can see the if and else parts of the steps. The way that *setPSPBackground()* works is that if it can detect that we have set up a layer named "Horizontal" in the template, it will set the visibility of the appropriate layer to the orientation of our artwork that we put in the UI. It only looks for "Horizontal" because if we have the options for both a horizontal and vertical layout background, "Horizontal" should exist no matter what. When setting up a template, the layers do need to be named "Horizontal" and "Vertical" without quotes, and it is case sensitive.

If we don't need to have different backgrounds based on the orientation of the background, our PSP background should just be the "Fill" layer. If we call *setPSPBackground()* and there is no "Horizontal" or "Vertical" layer, the function will just make sure that "Fill" is visible.



PSP DTS Magnet which needs rotation based on the art (left) vs PSP DS Mousepad which will always be a certain layout (right)

Do note that some PSP assets are in the Assets folder, including a tileable background image that you can plug into Photoshop's Patterns window. These might be helpful if you need to create a new PSP template.