**The Settings File**

Date: 8/11/2024

# 0. Before we start:

## 0.1 What this guide is for

This guide intends to act as a more complete reference for users of ImageFlow when configuring their pipeline settings file. This settings file is the JSON template you will use to set up where your input data is found, where your output will be stored, and what processing steps you want your pipeline to take.

## 0.2 What is used in this document

* [JSON](https://www.json.org/json-en.html): JavaScript Object Notation (JSON) is a popular way of formatting information in a way that is accessible to humans and easily processed by computers.

## 0.3 Helpful resources

* [JSON Lint](https://jsonlint.com/) A website that allows you to copy JSON directly into it and validate if it is formatted properly or not. This can be very useful in debugging if you are getting errors related to your settings file.
* [VSCode](https://code.visualstudio.com/) a popular, lightweight, and very well supported code editor that will present your settings file in an easy to engage with way.
* [VSCode SSH](https://code.visualstudio.com/docs/remote/ssh) a VSCode plugin that makes developing on a remote machine (such as the Euler) easy, allowing you to do everything through the easy to understand VSCode interface.
* [Ilastik Headless Mode Guide](https://www.ilastik.org/documentation/basics/headless) describes how to run Ilastik projects from the command line

# 1. Making a new Settings File

A template settings file is included in the settings directory of ImageFlow. You can copy this to start with a blank settings file when beginning a new project. You can also share settings files between colleagues if you have access to all of the resources it references.

# 2. Understanding the Settings

In this section we will step through each value in the settings file in order of appearance, describing what type of value it expects and what it does.

All paths to folders/files should be absolute unless specified otherwise.

## 2.1. input\_path

input\_path expects a string (value wrapped in “ “). If my images are stored in a folder called “images”, I should set input\_path to “path/to/images”. This value could also be set to a single image, for example, to use “pic1.tiff” in the images folder, input\_path would be set to “path/to/images/pic1.tiff”.

## 2.2. output\_path

output\_path expects a string. If I want my output stored in a folder called “analysis” I should set output\_path to “path/to/analysis”.

## 2.3. ilastik\_application\_path (only if using Ilastik)

ilastik\_application\_path expects a string. This string should define the pathto your system’s executable Ilastik application. Where this is will depend on what type of system you are using, however, it should end in “.sh” on Mac/Linux, and “.exe” on Windows.

## 2.4. ilastik\_classifiers\_path (only if using Ilastik)

ilastik\_classifiers\_path expects a string. This string should define the path to the folder containing the pretrained Ilastik Pixel Classifiers you intend to use in this pipeline.

## 2.5. cell\_profiler\_application\_path (only if using Cell Profiler)

cell\_profiler\_application\_path expects a string. The value of this string will depend on whether you are running your pipeline locally (in which case, set it to the Cell Profiler application executable on your device) or on Euler, which is more involved. On Euler, cell\_profiler\_application\_path must rull Cell Profiler with Singularity, so it should be set to “path/to/singularity run path/to/cellprofiler cellprofiler”.

## 2.6. cell\_profiler\_pipelines\_path (only if using Cell Profiler)

cell\_profiler\_pipelines\_path expects a string. This string should define the path to the folder containing the premade Cell Profiler pipelines you intend to use in this pipeline.

## 2.7. EXPERIMENTAL: fiji\_application\_path (only if using FIJI)

fiji\_application\_path expects a string. This string should define the pathto your system’s executable fiji application. Where this is will depend on what type of system you are using, however, it should end in “.sh” on Mac/Linux, and “.exe” on Windows.

Note that FIJI support is still minimal.

## 2.8. EXPERIMENTAL: fiji\_macros\_path (only if using FIJI)

fiji\_macros\_path expects a string. This string should define the path to the folder containing the premade FIJI macros you intend to use in this pipeline.

Note that FIJI support is still minimal.

## 2.9. input\_channels

Input\_channels expect a dictionary (a list of additional values encapsulated in curly braces {}). This dictionary should describe the channels present in your input images in the order they appear. Currently, the system also depends on your image names having distinct identifiers of these channels, separated by underscores (\_). For an image with two channels, one imaging 561-TMRE and the other 488-SYBR, it should be named “anything\_561-TMRE\_488-Sybr\_anything.tiff” (although, the order of the channels in the file name does not matter, just the order in the setting). In the input\_channels dictionary we define these channels and how we can identify them later on in the pipeline. So, in our example, we can set input channels to:

{“TMRE”: “561-TMRE”, “SYBR”: “488-SYBR”}

The key values in this dictionary (TMRE and SYBR) will be used to identify available Ilastik classifiers for this image channel, if using Ilastik classifiers, so make sure they appear in the classifier names.

## 2.10. pipeline and pipeline.params

pipeline expects a dictionary. This dictionary will describe the steps you wish for your pipeline to perform on your dataset. Each step is a string value with a dictionary of optional settings. Further, each step also contains a params dictionary that can be populated with optional values that are relevant to the pipeline step they are contained in. NOTE: by default, you can leave params empty (“params”: {})

The currently supported pipeline steps, along with their possible params settings are:

* separateChannels: separates multi-channel images into distinct images
  + skip: expects a Boolean (true or false) of whether or not to skip this step when running
* runIlastikClassifier: runs a single or a set of Ilastik classifiers
  + skip: expects a Boolean (true or false) of whether or not to skip this step when running
  + classifiers: expects a list of strings containing the names of the Ilastik Classifiers in your ilastik\_classifiers\_path you want to run. Must contain the “.ilp” extension.
  + “--" flag options. Experimental flag settings you can add to Ilastik headless commands to have more fine-tuned control over the pipeline. See the [Ilastik Headless Mode Guide](https://www.ilastik.org/documentation/basics/headless) for info on how these work.
* runCellProfilerPipeline: runs a single or a set of Cell Profiler pipelines
  + skip: expects a Boolean (true or false) of whether or not to skip this step when running
  + pipelines: expects a list of strings containing the names of the Cell Profiler pipelines in your cell\_profiler\_pipelines\_path you want to run. Must contain the “.cppipe” extension.
* runFijiMacro: runs a single or a set of FIJI macros -- *EXPERIMENTAL*
  + skip: expects a Boolean (true or false) of whether or not to skip this step when running
  + macros: expects a list of strings containing the names of the FIJI macros in your fiji\_macros\_path you want to run.