

Testing Pipeline Stages

from: [DC2 Management -> Running the Pipeline at NCSA](#).

This page is meant to help developers test pipeline stages.

The first step, of course, is to write your pipeline stage. Read [CreateStageImplementation](#) for help. But after that, how do you exercise it?

For testing purposes you will probably want to run a mini pipeline that has just three stages: an input stage (`lsst.dps.IOStage.InputStage`), your stage and an output stage (`lsst.dps.IOStage.OutputStage`). The input stage will serialize the objects needed as input to your stage, similarly for output. [PersistenceIoStages](#) describes input and output stages.

Pick a directory to contain the pipeline execution shell script and configuration files. An example is available in `$DPS_DIR/bin`. Your directory should contain a copy of `nodelist.scr` and `run.sh` and a policy directory for policy files:

You will need to write one pipeline policy file (see [PolicyFormats](#)) plus a policy file for each stage of your test pipeline. The pipeline policy file lists the various stages and the policy files for each. For example your `pipeline_policy.json` might contain:

```
{
  "appStages": [ "lsst.dps.IOStage.InputStage", "...yourstage", "lsst.dps.IOStage.OutputStage"
  "stagePolicies": [ "policy/input_policy.json", "policy/yourstage_policy.json", "policy/output_policy.json" ]
}
```

Configure the pipeline files `nodelist.scr` and `run.sh`. To run a single node `nodelist.scr` must have a `:2` after the machine name and `run.sh` must specify `nodes=1` and `nslices=1`.

Add file `.mpd.conf` to the user home directory of each node on which you will be running. It must have mode 400 and contain `MPD_SECRETWORD=anyword` where *anyword* is the same for all nodes.

Read [RunPipelineHarness](#) for more information on running a pipeline.

Examples

See `imageproc/pipeline/examples` for two example pipelines. `runCopyPipeline.py` runs a trivial pipeline stage that simply copies a [MaskedImage](#) from input to output. `runSubtractPipeline.py` runs the image subtraction stage. The code for the stages is in the [usual location](#): `python/lsst/imageproc/pipeline`

To test the image subtraction pipeline stage with an arbitrary pair of input images I am writing template versions of policy files that use python `%(name)s` substitution. A simple shell script will fill in these templates using command-line arguments, write out the resulting file and run the pipeline.

The preprocess, process and postprocess methods

The basic idea is:

- preprocess and postprocess run on a single instance of your stage. Thus preprocess can write instance variables that postprocess can read.
- The process stage is run in one or more separate instances of your stage class (one per slice). The preprocess and postprocess methods are never run in these instances.

But how does data sharing work for preprocess -> the slices -> postprocess? Apparently there are some details yet to be worked out on that. (In fact there is no use case for preprocess; it was added for symmetry with postprocess, for which one use case is writing provenance information).

Logging

Presumably there is some way to see logged messages as the pipeline runs. I have not yet looked into how to do it.