

Civet: Option Override Processing

## Background

Civet tool description files contain tool options and parameters. For tool XML reusability between pipelines and to facilitate research use by allowing experimentation with tool parameters, it is important for the Civet framework to provide a mechanism for both pipeline developers and research users to be able to override an option’s default value.

For clinical use (such as CGA), only pipeline developers will be able to specify alternate values for a tool option.

## High-level design

The mechanism used for overriding tool options is through an override file. This override file contains a list of options and their value (exact format to be described in a later section). These values are substituted for the value specified for that option in the tool definition XML.

A pipeline may have its own option override file contained in the same directory as the pipeline description XML file. This override file is has the same filename as the pipeline definition file, except the extension is “.options” instead of “.xml”. This facilitates tool reuse by allowing a pipeline developer to specify different options for a tool. Without option overriding, a tool developer would need to create a new tool XML file just to change a single command line parameter.

For research use a Civet user may pass his or her own option override file to the civet\_run command using the --option-file (-o) command line switch. User-specified option overrides take precedence over pipeline-level option overrides, so if the same option is overridden by the pipeline developer and by the pipeline user the pipeline user’s value is used.

## Design Details

### Tool options

The Civet XML spec defines an <option> tag that can be used in a tool description. Command line parameters that are specified as an <option> can have their value pulled from a file, from a value given in the XML, or from an override file.

<option name=”...” from\_file=”...” command\_text=”...” binary=”...  
 threads=”...” value=”...” />

Note that the from\_file , threads, and value attributes are mutually exclusive and cannot be specified for the same option tag. Currently options with the from\_file attribute cannot be overridden; only the value or threads attribute of an option can be overridden. The binary attribute is combined with the value attribute and indicates that the value can be True or False. If the value is true, the command\_text will be used to substitute for the option in the command line. If the value is false then an empty string will be substituted for the option in the command line.

If a threads=”True” option is overridden, then this may override the Tool’s threads attribute when submitting the job. (If all threads options are overridden, then the maximum value for any thread option will be used as the ppn value during job submission.)

Option names are in the same namespace as tool file IDs, and can be used in the command in the same way tool IDs can.

For example, this option:

<option name=”foo” command\_text=”-f” value=”10” />

would substitute -f 10 for {foo} in the tool command line.

### Option File Format

Since each tool has its own namespace for file IDs and option names, we need a way to prevent collisions with option names in the pipeline’s option override file (since multiple tools in the pipeline may have options with the same name). The solution is to introduce a prefix used for naming the options in the option override file. A tool definition has a tool\_config\_prefix attribute. This prefix will be used for specifying the namespace the option belongs to in the file. Here is an example of tool\_config\_prefix:

<tool name="BWA\_Alignment"

threads="16"

walltime="20:00:00"

tool\_config\_prefix="bwa\_aln"

error\_strings="'Abort!'">

…

</tool>

The file consists of one option per line, in the format tool\_config\_prefix.option\_name=value. Lines beginning with the # character will be ignored.