Passing parameters to tools

How to pass parameters to tool

How to pass parameters to tools This document proposes an updated method of passing parameters to tools when invoking them. We used to do this another way. The intent of this proposal is to make it work properly across all tools on all hubs. Step 1: A tool is launched with various parameters as an argument a) Parameters are defined with a simple syntax that captures the parameter type, an optional parameter name, and its value: directory(context):/tmp file(input):/data/groups/testgroup/dropbox/input.txt file:/home/neeshub/mmclennan/.bashrc Each parameter value has the form type(name):value, where type is either file or directory. We may add other types as time goes on, but weâ€™ll never add something general like string. Itâ€™s far too dangerous to let the application parse untrusted string values, so all types must be well-defined and validated by the middleware. b) The tool invocation URL takes a params field value with a newline-separated list of parameters. For example: https://nees.org/tools/indeed/invoke/current?params=directory%28context%29%3a%2ftmp%0a%0dfile%28input%29%3a%2fdata%2fgroups%2ftestgroup%2fdropbox%2finput.txt As you can see, the ()â€˜s, :â€˜s, and other punctuation characters are encoded to the URL query notation. But the original text before the encoding for this example was quite simple: directory(context):/tmp file(input):/data/groups/testgroup/dropbox/input.txt Just like the original example, but only the first two parameters. Note the separator characters %0a%0d in the URL. These are good, since theyâ€™ll never conflict with other syntax and they mimic the syntax that we eventually get in the parameter file. Step 2: The web server receives and validates the information The web server examines the params field and parses the syntax for all elements. It scans through and validates all elements, checking their type and value. For file and directory types, for example, the given file path must reside in a white-listed set of known places, including the userâ€™s home directory, the /data directory, the /scratch directory on NEES.org, etc. Files and directories must also exist, so that a malicious hacker canâ€™t try to pass in shell commands in place of a file. For file and directory types, the given file path must reside in a white-listed set of known places. For NEES.org, the whitelist will include the /home and /nees directories. The web server will also perform small translations on the params string, like collapsing and stripping extra newlines. If a value is bad, or an argument type is not known, the web server should halt processing and display an error. The web server will not check for the existence of the file or directory because it does not have complete access to the directories and the intended use of the parameter is unknown. Parameters could specify output file names, in which case the file would not exist. File validation is is the responsibilitiy of the application. Step 3: The middleware is told to start a session with the arguments The middleware already has an option for appopts. Weâ€™ll leave that alone for backward compatibility and create a similar params option. The params option will receive the URL-encoded params string from the web server. It will decode it using â€˜urllib2.unquote(params).decode(â€œutf8â€•)â€™ and write it to a file called parameters.hz in the userâ€™s session directory. It will also set an environment variable TOOL_PARAMETERS=parameters.hz within the session indicating to the tool that parameter file exists. Thatâ€™s all the middleware needs to do. No need to pass any parameters into the invoke script. The tool (or perhaps the invoke_app wrapper) will take it from there. Step 4: The tool is invoked. Many tools will use the new invoke_app script to look for arguments and pass along appropriate arguments. For those like inDEED that may want to handle the arguments themselves, they can. They would simply look for the $TOOL_PARAMETERS environment variable. If set, it points to a file containing the sanitized tool arguments in the form shown earlier: directory(context):/tmp file(input):/data/groups/testgroup/dropbox/input.txt file:/home/neeshub/mmclennan/.bashrc The tool would then read this file and parse the various lines to extract arguments.