

# ToolSelector v1.0 User Guide

---

*EnSoft Corp. / 2011-10-25*

## Contents

Tool Overview .....	2
Purpose .....	2
Features .....	2
Concepts .....	2
Operation .....	2
Profile Configuration .....	4
General Information .....	4
Rule Specifications .....	4
File Extension .....	4
Argument Pattern .....	5

## Tool Overview

### Purpose

ToolSelector is a utility program that can be used to “select” between one or more configured tools based on certain properties of the input arguments to ToolSelector. It is similar to a batch or shell script in this respect, but can be configured more easily and with more advanced logic. The primary use case that it was designed for is to select between multiple diff/merge tools based on the file types of the files to be diffed/merged. Many repository systems support an external diff/merge tool, but only one tool can be configured at a time. ToolSelector provides a convenient way to configure multiple diff/merge tools for use with such a system.

### Features

- ToolSelector is highly configurable, and can be launched using multiple configured **profiles**. This allows it to be used simultaneously with multiple repository system and tool configurations.
- ToolSelector can remap arguments from the repository to arguments acceptable to the diff/merge tools, on a per-tool basis.
- ToolSelector can remap return codes from the diff/merge tools to return codes acceptable to the repository system, on a per-tool basis.
- ToolSelector comes with predefined **rules** that make it easy to specify logic for selecting one tool or another based on file extension or other argument properties

### Concepts

ToolSelector is designed to work flexibly with a variety of tools and systems. The logic for a particular combination of source system and selectable tools is specified in a ToolSelector **profile**. Profiles are saved as configuration files in the same directory as ‘toolselector.exe’; the file name (less extension) is called the **profile ID**.

Various configuration parameters are stored inside the profile configuration file. There is a global parameter that defines the **tool priority** – a list of tools to select between, and the order in which to evaluate them. Most of the parameters define the properties of one or more **tool profiles**. Each tool profile has a **tool ID**, as well as a tool path, mapping rules for arguments and return codes, and a set of **rules** that are used in deciding whether or not to use a tool. The parameters that can be configured for each rule depend on the **rule type**.

### Operation

Basic usage information (this can also be queried by running ‘toolselector /?’ on the command-line)

```
Usage: ToolSelector -profile=<profile_id> [arg0 [arg1 [...]]]
```

```
-profile - identifies the profile to load
argN     - optional command-line argument(s) to pass to the selected tool
```

When ToolSelector starts, it loads the tool profile specified as the first command-line argument. All subsequent arguments are treated as **input arguments** that can be passed along to the selected tool. After loading the profile, ToolSelector will evaluate each tool profile in priority order. A tool “passes” evaluation if none of its rule requirements fail to be met. The first tool that passes its evaluation is selected by ToolSelector. (Note – a tool with no rule requirements is always selected when evaluated. It is recommended to configure a ‘default’ tool with no rule requirements at the end of the priority list.)

Once the appropriate tool is selected, ToolSelector will launch the tool, providing it with input arguments as specified in the tool profile. ToolSelector will then wait for the selected tool to complete execution, and capture the return code. If any return code remapping rules are specified, ToolSelector will adjust the return code according to those rules before exiting with that return code.

## Profile Configuration

### General Information

Profiles are stored with file extension 'ini', but they follow the serial specification for the Java Properties class. Please see the Java documentation for more information about escape characters and other special formatting.

Some parameters allow an **argument range** to be specified. An argument range is specified by the following format:

`<start-index>..<end-index>`

Example: `0..3`

The range is inclusive (both argument 0 and argument 3 are included in the range above). A single argument index can also be specified, in which case it is treated as a range between that number and itself (i.e. the value '1' is treated as a range '1..1')

### Rule Specifications

All rule parameters are mandatory unless otherwise specified.

#### File Extension

This rule's requirement is met if a configured subset of the input arguments representing file paths all end with the same file extension

Type ID: `file-extension`

Parameters:

- `extension` - file extension that all specified arguments must end with
- `inputargs` - input arguments to check, specified as a comma-separated list of argument ranges.

Example:

```
mytool.rule.foo = file-extension
mytool.rule.foo.extension = txt
mytool.rule.foo.inputargs = 0..1, 3, 5..6
```

Example results:

- [PASS] C:\myfile.txt
- [PASS] C:\repository\myfile.log.txt
- [FAIL] C:\repository\myfile.log
- [FAIL] C:\temp\myfile.txt.0

## Argument Pattern

Type ID: `arg-pattern`

Arguments:

- `pattern` - a regex pattern to check against, specified in a way that it can be understood by the Java Pattern class. Don't forget to account for the properties file formatting as well.
- `inputargs` - same as for the File Extension rule (see above)

Example:

```
mytool.rule.foo = arg-pattern
mytool.rule.foo.pattern = \\\.mdl\\.\\.\\d+$
mytool.rule.foo.inputargs = 0, 1, 2
```

Example results:

- [PASS] C:\temp\myfile.txt.0
- [PASS] C:\temp\myfile-2.txt.004
- [FAIL] C:\myfile.txt
- [FAIL] C:\repository\myfile.log
- [FAIL] C:\temp\myfile-2.004.txt