Table of Contents

Introduction

Tools provided by the Stand-Alone Tool Kit (SATK) vary in complexity. Some have only a few user supplied options, others have many with complex relationships. ASMA is an example of a complex tool. The configuration system described in this manual provides a standard approach intended to simplify the use of such tools.

This manual describes a configuration system that unifies user supplied inputs for invoking a Python-based tool. A simple tool allowing management and testing of the system is documented in this manual.

Specific tool documentation utilizing the configuration system will use terminology and concepts introduced in this manual. Facilities described in this manual will be available to such tools aiding the use of the overall system.

Getting Started

Step 1 – Review this manual

In particular review sections “Configuration System Structure” and “Using a Tool with the Configuration System”.

Step 2 – Determine site directory location and desired current working directory path behavior.

Step 3 – Initialize the configuration file system using `config.py`. Refer to the “--init” section for a description of the process.

Step 4 – Test the configuration system

Use: `config.py --tool config --cinfo L afile`

If you need to redo either use of `config.py` in Step 3, you may need to use the `--force` option to overwrite an existing file.

At this point you should be able to use any SATK tool for which user configuration files are supplied and begin using your own. Review other portions of the manual as appropriate.

Configuration System Structure

The INI format files that compose the configuration system must reside in some directory. Locating where a configuration file resides is part of the system's role. The configuration system depends upon a configuration site directory. It is identified by either of two mechanisms:

- an environment variable, `SATKCFG`, or
- a `SITE` section containing the option `satkcfg` in the `site.cfg` file.

The `site.cfg` file has a fixed location, namely the `config` directory within the SATK directory.

site.cfg

The `site.cfg` configuration file provides option information required for processing of configuration file found elsewhere.

Only two such options are currently supported:

- `cwduse` – the option defining how the current working directory is handled in search order path creation, and
- `satkcfg` – the “link” to the user's site directory.

The `site.cfg` configuration file may be initialized using the configuration tool, `config.py`, described below. If a `site.cfg` file is not supplied, then the `SATKCFG` environment variable is required for use of the configuration system. If both are provided, `SATKCFG`, being an explicitly specified variable, will take precedence over the `satkcfg` option in the `site.cfg` file. `site.cfg` effectively provides a “link” to the user's site directory.

If neither the `SATKCFG` environment variable nor the `satkcfg` option are present, the configuration system is effectively disabled. All run-time options must be supplied via explicit environment variable specifications and command-line script arguments. Tools operate as they did prior to availability of the configuration system.

The Site Directory

The site configuration directory provides a location outside of the SATK directory for the storing of site specific configuration files. This includes a configuration file for each tool defining the tool's global defaults and user supplied configurations for use with specific tool input sources.

When the `config.py` tool initializes the site directory, it will place within it a file for each tool using the configuration system. The file's name will be *`tool`*`.cfg` where *`tool`* is the name

by which the system recognizes the tool. For example, the `config.py` tool will create a `config.cfg` file for itself. The tool specific file will be described as a “tool configuration.”

Tool Configuration Files

As initialized, the tool specific configuration file will contain at least one section, `DEFAULT`. As initialized, all of the options that have defaults will be commented out. They are provided for the user to change the defaults by altering the option's value and removing the initial comment character.

The optional `CONFIGS` section within the tool configuration file is reserved for user configuration files. The `CONFIGS` section provides “links” to user configuration files located anywhere in the file system based upon a name. See the “User Configuration Files” section, following for details.

User Configuration Files

The user may elect to group different uses of a tool in a common file. A tool's identified source for a given usage specifies a section within the configuration system devoted to that source's options. This section may be placed in the tool configuration file or another file. Technically, such sections may also be placed in the `site.cfg` file, but this is not recommended.

All tools using the configuration system support the optional command-line argument `--cfg`. This argument identifies the additional user configuration file to be used by the tool. If this file does not reside in the site directory, then an entry for it is required in the tool's `CONFIGS` section. The file identification option may be a file name or some other string and its value is the absolute path to the specific user configuration file. If the option is not found in the `CONFIGS` section, the site directory is assumed.

For the purpose of assembling sample or test programs, ASMA provides a user configuration file and initializes its `asma.cfg` tool configuration file's `CONFIGS` section to locate them.

Using a Tool With the Configuration System

All SATK tools are invoked from a command-line or user developed script on the user's platform. Most tools require command-line Python script arguments. Usually defaults are available. More complex tools may require various environment variables for proper execution of the tool. Most environment variables are used to specify a directory search order for the location of files used by the tool. This approach results in two sources of information for use of the tool with two tiers of information:

- explicitly specified environment variables or a default if omitted, and
- explicitly specified tool command-line arguments or an argument's default if omitted and available.

The configuration system adds a third tier that sitting between explicit and default values. Explicit specification of a value always takes precedence when a tool is invoked. If explicit values are not provided, the configuration system supplies the value for a given environment variable or command line argument. As a last resort, a default is used when available. Values are selected in high-to-low priority order:

1. explicit values (command-line arguments or environment variables),
2. configuration system values (explicit or defaulted),
3. tool defaults (for omitted options).

All of these values are considered an “option” in the configuration system. Option values may have configuration system supplied defaults for omitted configuration file options. These override tool built-in defaults, thereby allowing a user's override to a tool's built-in default.

Configuration System Command-Line Arguments

Four command-line arguments are available to tools using the configuration, only one of which is required. “CL” column indicates whether the option is available in the command line. “CFG” indicates whether the option is available in a configuration file.

Argument	Required	Values	Type	Default	CL	CFG	Description
`--cfg`	no	single	FILENAME	none	yes	no	Identifies a user configuration file
`--cinfo`	no	single	See desc.	none	yes	yes	Provides configuration system information
`--help`	no	no	EFLAG	False	yes	no	Provide command-line help
positional	yes	single	SOURCE	none	yes	no	Positional argument identifying tool input

None of these options may be specified by environment variable.

--cfg

The `--cfg` command-line argument identifies a user configuration file to be included in the configuration processing of the tool. If the tool's configuration file contains a value for this command-line argument in its `CONFIGS` section, it will use the FILEPATH it finds to locate the user configuration file. Otherwise, the command-line argument is assumed to be a FILENAME existing in the user's site directory.

--cinfo

The `--cinfo` command-line argument provides run-time information to sysout related to various aspects of the configuration system. Various types of information can be presented depending upon the set of single character options provided in the option's value. All requested options are specified in a sequence of characters without any intervening white space. The information is presented in the sequence in which the characters are specified.

The information represents the results of the configuration process.

The following information types are available:

Type	Description
`A`	Script arguments from the command-line as seen by the tool
`D`	Configuration file defaults.
`E`	Tool related environment variables.
`F`	Configuration files read by the configuration system for the tool's execution
`L`	All configuration options, equivalent to `AYEVUFSDO`.
`O`	Selected configuration option values and each option's source
`S`	Configuration file sections found
`U`	Tool's usage of configuration options including built-in defaults.
`V`	All environment variables available to the tool (includes those provided by type `E`)
`Y`	Python's `PYTHONPATH` as used by the tool (includes dynamically added directories)

Some information may be summarized using various character codes. The data will include keys explaining the codes.

Options `AYEV` are useful in understanding the external information available to a tool's execution. Options `V` and `Y` should rarely be needed.

Option `U` describes how the tool uses various configuration options and from which sources (command-line, environment variable or configuration file) it may be used. Defaults when available are included. This information does not change from one use of the tool to the next, but is helpful in understanding the options ultimately presented to the tool.

Options `SDO` describes the configuration information provided to the tool itself for its run-time options.

-h/--help

Enables command line help information. After the help information is provided, the tool terminates.

positional

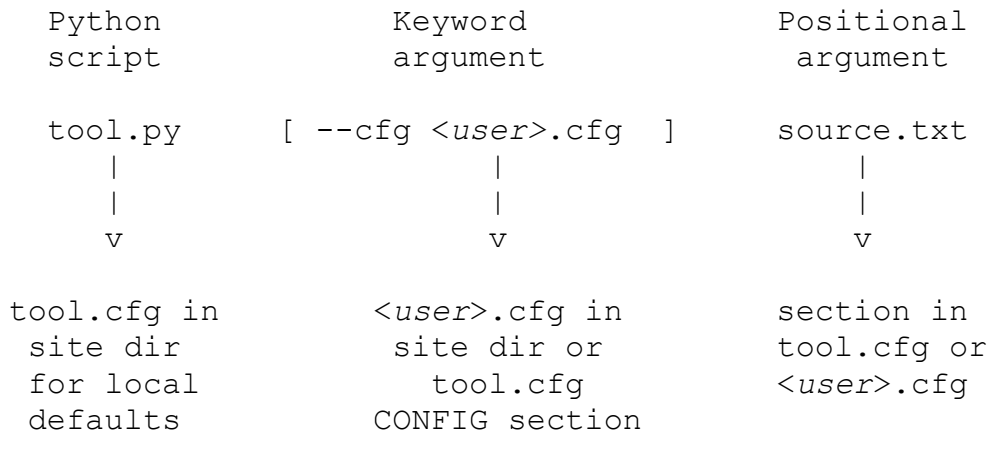
The command-line positional argument is of type SOURCE. It identifies the configuration file section that contains the configuration file options for use by the tool for this input source. The section, if found, must be defined in the tool's configuration file or the configuration file specified by the `--cfg` command-line argument.

Technically the section could be placed in the `site.cfg` file. But, it is recommended that local configuration information reside in the local site directory outside of the SATK directory.

If the identified section provides a value to the configuration file option used for identification of the input file, it will be used as the input to the tool. Otherwise the value supplied by the positional argument is used directly as the input. In either case (the positional argument or the input file option) must result in a FILENAME value. Any directory search order is used with the resulting FILENAME value.

Command-line and Configuration File Relationships

The command-line when using the configuration system directly relates to the configured files and sections used. This diagram illustrates the relationships. `<user>` indicates a user supplied name.



In each of the two command-line arguments, a section may be identified:

1. In the case of the `--cfg` argument, when the actual FILENAME is specified in the `CONFIGS` section of the *`tool`*`.cfg` file and
2. When the configuration input option identifies the actual FILENAME.

In either case an actual file name with an extension is not required. As long as the value supplied in the command line matches the `CONFIGS` section option in the first case or source specific section in the second, the value may be used.

However, if the value identifies an actual file, then it must conform to usage on the platform of a FILENAME type option.

For sample tool input or test input, SATK may supply configuration files for their use within the SATK directory. The tool's `CONFIGS` section will be appropriately initialized in this case. Utilization of SATK supplied user configuration files requires initialization of the configuration file system by the user.

Run-Time Tool Information

Options for SATK Python tools utilize two fundamental mechanisms for providing run-time related information:

- command-line Python script arguments and
- platform environment variables.

Each source of run-time information has equal priority because no overlap has exists. For a given piece of run-time information either a command-line argument is used or an environment variable. A built-in default might exist for the option. Use of these two source requires some form of user supplied platform specific script to establish environment variables and the necessary command-line arguments.

The configuration system described in this manual introduces a third, entirely optional, source of run-time information:

- a text file using the common INI file format.

This new source, a configuration file, allows options to be specified within the file that would have been specified by either of the previous sources. Elimination of the need for platform specific scripts, particularly for complex tools, is the goal of this new run-time information source.

SATK has always taken the view that the information supplied closest to the point in time a tool is used takes precedence. The natural behavior of default values is an example of this strategy. A built-in default established at the time a tool is constructed is far removed from the point in time information is provided when a tool is actually used.

Information supplied by a user text file is certainly closer in time than a value established at the time a tool is developed. And information supplied when a tool is run is also closer in time than is a previously provided text file. Environment variables, while close in time to the point at which a tool is executed, must precede its actual use to be available to the tool. With these observations being made the SATK strategy results in the following priority of information sources:

1. command-line Python script arguments,
2. platform environment variables,
3. text file information, and
4. built-in defaults.

Previously, built-in defaults were explicitly applied to either an environment variable or a command-line script argument. In this hierarchy, a built-in default is only used in the absence of a higher priority information source.

It should not be inferred from this priority scheme that all information may be supplied by any

of the sources. Each tool will define which pieces of information are supported by which sources. The priority scheme only applies when run-time information is supported by the tool from multiple sources. Each tool's documentation will indicate for a given option how it is supported. The tool's functionality may drive the available sources.

Tool Options

Each option, regardless of source, has a set of common attributes:

- the number of values supported (no values, one value, or multiple),
- whether it is required or not,
- whether it has a built-in default or not, and
- from which sources the option may be supplied.

Each tool will document these attributes for its supported options.

While each source influences how options are supplied, the options themselves for a given type of information are consistent for all sources. How to specify options in each source is described first and then how option values representing different forms of information are described.

Option Names and Case

All three sources of options depend upon a “name” being used with the option. Upper case names are used by environment variables. Command-line arguments are typically lower case, although upper case may be used. Configuration files may use either upper or lower case names, but all names within configuration files are case insensitive and treated as lower case.

Regardless of the case of an option used by a given source, the system treats all names as lower case. Usually the same “name” is used for all sources, but different names used by different information sources is supported and may arise. Tool documentation will make such cases clear.

Command-line Arguments

All command-line options are provided following the identification of the Python script that performs the tool's functions. Like this...

```
tool.py option1 option2 ...
```

Each option is identified by a character sequence. The sequence may be upper or lower case. Options are separated by at least one space. Options may use a single-character short identifier or a long multiple character sequence. In some cases both options may be available. Short options are preceded by a single hyphen, `-`. Long options are preceded by two hyphens, `--`. In this example the short option uses the letter `s` and the long form uses

multiple characters.

```
tool.py -s --long
```

Options that begin with a hyphen are considered key-word arguments. Options that do not begin with a hyphen are positional arguments.

Options Without Values

An option that does not require a value, the short or long form of the option identifier is used simply without a value. The previous example illustrates two options, neither of which uses a value.

Single Option Value

An option using a single value has the value placed immediately following the option identifier:

```
tool.py --max 20 -n 15
```

Multiple Option Values

For an option in the command-line for which multiple values are used, the option is simply placed multiple times in the command-line in no particular order:

```
tool.py --list 5 -m 7 --list 6
```

In this example, two options have been supplied:

- a single value option, `m`, whose value is `7`, and
- a multiple value option, `list`, whose values are `5` and `6`.

Command-line Help

All tools will provide a command-line help option of `-h` or `--help`. When either of these options are supplied in the command-line, the help information will be provided to the command-line and further processing will cease.

Command-line help indicates optional arguments by enclosing them in square brackets. Because configuration options may have other sources, only options that truly must be supplied in the command-line will be indicated as such, meaning without square brackets.

This does not change the overall requirement for an option. It simply means that its optional in the command-line. The tool will fail to launch if an option required in the command-line is not provided.

Environment Variables

Each platform has its specific mechanism for establishing an environment variable's value. However this does, the Python tool can access its assigned character string. SATK tools do not use environment variables for which no value is assigned. SATK usually restricts use of environment variables to directory related information. For the purpose of illustration, the examples will use Linux examples.

Single Option Value

Make the assigned environment variable available to the tool by specifying a string. The entire string is interpreted as a single value.

```
export USE_DIR=/home/user/use/this/one
```

This example assigns an absolute directory path to the variable `USE_DIR`.

Multiple Option Values

SATK uses multiple option values assigned to an environment variable for specification of directory search order paths. This use of such paths will utilize a platform specific separator for each directory. The platform specific separator is honored.

```
export PATHS=/home/user/dir1:/home/user/dir2
```

Here multiple values, the two specified directories are provided to the tool for searching for a file.

Configuration File Options

Without understanding how a configuration file in general is constructed, description of how to specify options within the file lacks context. The next section describes the configuration file in general, before the related discussion of options.

Configuration File Structure

Each configuration file follows the INI file format. INI file usage has multiple implementations and they usually vary in capability. Other than the most general aspects of the format, no standard exists.

In the case of SATK, a Python library module is used for configuration file parsing. For official documentation of the module, refer to the Python Library Reference documentation for the `configparser` module at https://docs.python.org/3/. This is section 14.2 in the “Library Reference” link on the page.

Each configuration file contains one or more sections. A more familiar term for some will be “stanza”. Python calls them “sections.” For consistency with Python documentation and internal source code comments, this document will use the term Python uses, namely, “section.”

A section without options is allowed. All options defined in a `DEFAULT` section regardless of the file itself will be merged as a set of global defaults used by the processor. Default options may be used for interpolation. Each section is enclosed in square brackets. A configuration file lacking sections is an error condition.

Section names used by the configuration system do not use embedded spaces and are always upper case.

```
[SECTION]
```

but user supplied section names may include embedded spaces and are case sensitive:

```
[ Section 1 ]
```

The first section's name is `SECTION`. The second section's name is `Section 1`, the spaces at the beginning and end of the name being ignored. Section designations may be indented for readability.

Within each section, options, sometimes referred to as “keys” are defined. Options are identified within a section by name. Option names are case insensitive, being treated as lower-case. If an option value is supplied, the option name is followed by a separator, being either a semi-colon, `:`, or an equal sign, `=`. Multiple values may be supplied following the separator on multiple lines. Options without values are allowed. Whether multiple values or no value are allowed or not is dependent upon the specific option's usage by its tool. Option names may be indented and their multiple line values **must** be indented more than the name.

Comments are indicated by either a pound sign, `#`, or a semi-colon, `;`. A comment may start anywhere in the text file line, making the entire line a comment. Or, a comment may follow the value(s) specified in an option line, allowing comments to be supplied for the option. Neither a pound sign nor a semi-colon may be used within an option line, signaling the start of a comment. Quoted strings and embedded spaces, for example, in directory or file names, are not allowed.

Additional lines may be used for an option's value provided the additional lines are indented further than the option to which it belongs.

Here are some examples of valid lines within a configuration file section:

```
[MYSECTION]
# Comments may be placed here
    ; or indented
option1: a_value
option2 = multiple values on a line or  ; and comments here
   continued on a another line          # or here.
```

These are general rules for how files are written. The configuration system works within these rules, supporting various types of options. The use and treatment of multiple values and multiple lines varies with the type of option.

Specifying Options in a Configuration File

Regardless of the number of values provided for an option within a configuration file, each option resides on its own line within the file. Unlike command-line arguments that may utilize two different character sequences for an option, the configuration files only supports one. The examples below omit the required section designation, but in a real configuration file, options do not exist outside of a section.

Options Without Values

Place the option as the first text in its own line, followed by an option delimiter. Using the same example from the command-line arguments section in a configuration file would result in:

```
s:
long:
```

Single Option Value

Provide a single value to an option in a configuration file by specifying the option's name, then a option delimiter followed by the value. Each option must be in its own text file line.

For this example, the preceding examples from from the command-line arguments section and environment variable are illustrated:

```
max: 20
n=15
use_dir: /home/user/use/this/one
```

This illustrates how the environment variable becomes lower-case within the configuration file and the two different option delimiters supported for configuration files.

Multiple Option Values

The option must be initiated with its own test line, followed by an option delimiter. Each of the values may be specified on the same or different lines. If multiple lines are used, the additional lines must all be indented further than is the option. Again, the preceding examples will be combined to illustrate their use within a configuration file.

```
m=7
list: 5 6
paths:  /home/user/dir1
    /home/user/dir2
```

Role of Configuration File Defaults

Configuration file defaults operate exclusively within the context of options supplied by configuration files. If the option is supported within a configuration file and no explicit value has been assigned to the option, but a value has been specified in a `DEFAULT` section, the default section value will be used.

Because the default value comes from a configuration file it takes precedence over a built-in default. This allows a user to specify a new default for the value. In essence another source is inserted in priority lower than an explicit configuration file option and higher than a built-in default.

Supplied Configuration File Defaults

The configuration system automatically supplies the following defaults available to

configuration files:

Supplied Default	Description
cwd	The current working directory when the tool was launched
cwduse	Current working directory default of 'omit'
date	The date in YYYYMMDD format
home	The value of the HOME environment variable if available
satk	The root directory of the Stand-Alone Tool Kit in which the tool resides
time	Time of day in the format HHMMSS using a 24-hour clock

If the tool utilizes an option of type SOURCE, this list will also contain a default value for its paired option identifying the actual input file used by the source.

These defaults, as such, may not be overridden with a different default value although they may be explicitly set, as with any configuration file default.

Value Interpolation

A value can be constructed from other supplied values by use of the interpolation syntax. To access another value defined in the same section, place the name of the option within curly braces, {}, preceded by a dollar sign, $.

```
satkdir = /home/user/SATK
filename: ${satkdir}/samples/xyz
```

This results in the option filename being set to the value:

```
/home/user/SATK/samples/xyz.
```

A value from a section defined in the same or a different file can also be referenced:

```
[PATH]
satkdir=/home/user/SATK

[mysample]
filename: ${path:satkdir}/samples/xyz
```

This has the same effect as the previous example in specifying the filename option's value, except that an option from a different section is used as the source of the interpolated portion of the value.

Any default may be used for interpolation purposes, including and especially those supplied by the configuration system.

Option Value Types

Previous sections focused on how to specify an option's value. This section describes different value types and how they influence the run-time configuration. While the overall effect is usually the same regardless of the source of the option's value, different sources may operate differently.

The upper case designation is the standard descriptor used for each option type when a tool document's its usage.

CHOICES – Value List

An option of type CHOICES accepts a specific set of values. The option may or may not have a default and may be required. Refer to the documentation of the option for the supported values and interpretation by the tool.

DFLAG - Disable Flag

A disable flag does what the name suggests. When a disable flag is present, it “disables” or “turns off” whatever is associated with the flag.

In the command-line, place the option with its leading hyphen or hyphens anywhere within the arguments to disable the option. The absence of the flag applies its default action, namely, allowing the option to remain enabled or turn on.

In a configuration file, when a disable flag does not have a value, it operates exactly as the option does in a command-line. The presence of the option disables it and its absence leaves it enabled. In a configuration file, a disable flag may also operate as a toggle switch when provided with a single value. See the “Toggle Switch” description for details. Each disable flag option requires its own line in the configuration file.

Disable flags are not used with environment variables.

DIR - Directory

When an option defines a directory value, it must be an absolute path. A DIR option is always single valued.

EFLAG - Enable Flag

An enable flag does what the name suggests. When an enable flag is present, it “enables” or “turns on” whatever is associated with the flag.

In the command-line, anywhere in the command-line code the option with its leading hyphen or hyphens. The absence of the flag applies its default, namely leaving the option disabled or “turned off”.

In a configuration file, when an enable flag does not have a value, it operates exactly as the option does in a command-line. The presence of the option enables it and its absence leaves

it disabled. In a configuration file, an enable flag may also operate as a toggle switch when provided with a single value. See the “Toggle Switch” description for details. Each enable flag option requires its own line in the configuration file.

Enable flags are not used with environment variables.

FILENAME – File Name

A file name option is a single valued option. It is typically used in conjunction with a PATH option. Technically a file name option is really a relative path. If a relative path is supplied it will be applied to the directory search order.

An absolute path may also be supplied for the option. In that case, any implied directory search order (PATH) normally associated with the FILENAME option will be ignored and the absolute path takes precedence.

File Name Extensions

All files supplied by SATK targeting cross-platform usage will use file name extensions, making them compatible with platforms requiring them. User developed or output files may follow the conventions of the user's platform.

The following user file name extensions are required regardless of the platform:

- `.cfg` – Configuration system text files.
- `.mac` – ASMA macro library definitions.

By convention these file name extensions are used:

- `.asm` – ASMA source and `COPY` directive files.
- `.msl` – ASMA Machine Specification Language files.

FILEPATH – File Path

The file path option type identifies an absolute path to a file.

PATH – Directory Search Order

A directory search order requires one or more absolute path (DIR) values. The sequence defines the order in which each directory is used for locating a file.

PATH options are usually available as environment variables and configuration file options. Most options use exclusion of option sources based upon priority. PATH options, while maintaining the priority sequence, use directories specified in the option's environment variable before searching any directories supplied by the option within a configuration file. This approach allows the two sources to be mixed. Simply refrain from mixing PATH options to exclusively use one source or the other.

Default values for a PATH option are only used in the absence of any explicit values supplied by an environment variable or configuration file option.

Whether and where within the search order the current working directory is placed is controlled by the global `cwduse` configuration file option. See the "–-cwduse" section for details. The option resides in the `DEFAULT` section of the `site.cfg` configuration file.

A PATH option is usually paired with a FILENAME option. See the FILENAME description of how the use of the PATH option is influenced by absolute paths supplied for a file name.

PATH options are not specified as command-line arguments.

SOURCE – Tool Input

Tool input is always identified by this single valued command-line argument. It is always a positional argument within the command-line. Any argument within the command-line that does not start with a hyphen (or two hyphens) is considered a positional argument.

The operation and specification varies depending upon whether the configuration system is in use. The configuration system is in use if the system can locate the SATK site directory. If the site directory can not be located, the configuration system is disabled and only environment variables and command-line arguments are used for tool run-time information.

Also, see the section "Using a Tool With the Configuration System".

Without Configuration Files

When the configuration system is unavailable, the tool input option within the command-line follows the rules of a FILENAME option and identifies the file providing the tool's processing input. See the "FILENAME" section for details.

In addition, the user must explicitly place the current working directory in any environment variable supplied search order path. This requirement results from control of current working directory behavior only being supported by use of configuration files

With the Configuration Files

The tool input option specifies a configuration system section that provides the options for processing the input file. It may follow the format of a FILENAME option or not. In either case, this option type identifies a configuration file section expected to be found in the configuration system. Whether the section is located or not, configuration file defaults will be recognized. If the section is located, explicitly defined options take precedence.

The actual input file is identified by the configuration file `input` option's FILENAME value (default or explicitly found). The `input` option's default value is always the value supplied by the tool input source command-line argument.

See the "Using a Tool With the Configuration System" section for details of these relationships.

Toggle Switch

Any enable (EFLAG) or disable (DFLAG) type option used in a configuration file may act as a toggle switch when coded with a value. The value is restricted to a set of choices.

Values which enable the option are:

`True`, `1`, `true`, `on`, or `enable`.

Values which disable the option are:

`False`, `0`, `false`, `off`, or `disable`.

When coded without a value, an EFLAF implies use of an enabling value and a DFLAG implies use of a disabling value

Regardless of whether the option is coded without a value (using it as a flag), or with a value (using it as a toggle switch), each option requires a configuration file text line of its own.

config.py – Managing and Testing the Configuration System

`config.py` is a Python tool, residing in the SATK `tools` directory, that allows the user to bootstrap the use of configuration files within the configuration system and perform some simple tests.

Two command-line arguments control the functions provided by the `config.py` tool:

- `--init` – initializes a portion of the configuration file system
- `--tool` – tests a tool's configuration processing.

If both are provided the initialization function occurs first and then a tool is tested. It is recommended the `--init` and `--tool` be used separately.

--init – Initializing the Configuration File System

Depending upon the state of the configuration file system, option `--init` may only partially depend upon their availability. Because this option initializes the files needed for using configuration files, they may or may not yet exist.

Because SATK itself provides configuration files, it is assumed the user will use configuration files at some point.

If the `site.cfg` file is not initialized any use of configuration files depends upon the `SATKCFG` environment variable for site directory identification. Using `site.cfg` is simpler than always establishing and environment variable, so it is assumed the user will prefer `site.cfg`.

--init satk – Initializing site.cfg

The `--init satk` option tells `config.py` to initialize the `site.cfg` file within the SATK `config` directory. If the `config` directory does not exist, it will be created.

Initialization of `site.cfg` is the first step in preparing the SATK for configuration file use.

The following additional options are used with `--init satk`.

Option	Required	Values	Type	Dft.	CL	ENV	CFG	Description
`--cwduse`	N	single	CHOICES	`omit`	Y	N	N	SITE section `cwduse` option
`--force`	N	no	EFLAG	`off`	Y	N	N	force overwriting existing `site.cfg` file
`--satkcfg`	N	single	DIR	none	Y	Y	N	SATK site directory location
`--verbose`	N	no	EFLAG	`off`	Y	N	Y	enables verbose output

Refer to the “Available Options” section for details.

--init site – Initializing the Site Directory

The `--init site` option tells `config.py` to initialize the site directory with tool configuration files.

This option depends upon:

- the ability to locate the site directory, for example, with `site.cfg` and
- the existence of the site directory.

This option does not create the site directory if it does not exist.

The following additional options may be used with `--init site`.

Option	Required	Values	Type	Dft.	CL	ENV	CFG	Description
`--force`	N	no	EFLAG	`off`	Y	N	N	force overwriting existing tool files
`--verbose`	N	no	EFLAG	`off`	Y	N	Y	enables verbose output

Refer to the “Available Options” section for details.

--tool – Testing the Configuration System

The `--tool` option causes `config.py` to use the configuration system with our without configuration files as if the identified tool is being launched. When using this option, `config.py` operates as any other tool would with the configuration system.

Once the configuration processing has completed successfully, this option attempts to open the input file using the `TSTPATH` and then closes the file.

This option, especially in conjunction with the `--cinfo` option aids in understanding how the system is functioning and why results are occurring as they are. Because the actual tool is not launched, but rather just the configuration processing used by the tool is performed, some situations may be more easily resolved.

As with any tool, modifications may be made to the `config.cfg` configuration file, the `config.py` tool configuration, once the site directory has been initialized. Such changes are only used with the `--tool` option.

Additional options available with the `--tool` option are:

Option	Required	Values	Type	Dft.	CL	ENV	CFG	Description
`--cfg`	N	single	FILENAME	none	Y	N	N	test with user configuration file
`--cinfo`	N	single	desc.	none	Y	N	Y	configuration system information
`input`	N	single	FILENAME	desc.	N	N	Y	Input file used with `source` section
`source`	N	single	SOURCE	none	Y	N	N	tested tool's input

Option	Required	Values	Type	Dft.	CL	ENV	CFG	Description
tstpath	N	multiple	PATH	desc.	N	Y	Y	a path option for testing
--verbose	N	no	EFLAG	off	Y	N	Y	enables verbose output

Refer to the “Available Options” section for details.

Available Options

This table summarizes the tool options available from `config.py`. The “CL” column indicates whether the option is available in the command line. The “Dft.” column identifies any available default. The “ENV” column indicates the option is available as an environment variable. The “CFG” column indicates whether the option is available in a configuration file. “desc.” means see the option description for more information.

Option	Required	Values	Type	Dft.	CL	ENV	CFG	Description
--cfg	N	single	FILENAME	none	Y	N	N	test with user configuration file
--cinfo	N	single	desc.	none	Y	N	Y	configuration system information
--cwduse	N	single	CHOICES	omit	Y	N	N	SITE section `cwduse` option
--force	N	no	EFLAG	off	Y	N	N	force overwriting existing files
--init	N	single	CHOICES	none	Y	N	N	initialize configuration system
input	N	single	FILENAME	desc.	N	N	Y	Input file used with `source` section
--satkcfg	N	single	DIR	none	Y	Y	N	SATK site directory location
source	N	single	SOURCE	none	Y	N	N	tested tool's input section
--tool	N	single	CHOICES	config	Y	N	N	tool configuration being tested
tstpath	N	multiple	PATH	desc.	N	Y	Y	a path option for testing
--verbose	N	no	EFLAG	off	Y	N	Y	enables verbose output

--cfg

Used with option: `--tool`

Type: FILENAME

This is the standard configuration system command-line option identifying the user's configuration file to be used with the tool. See the description of this option in the “Configuration System Command-Line Arguments” section.

--cinfo

Used with option: `--tool`

This is the standard configuration system command-line option identifying requested configuration info used by the tested tool. See the description of this option in the

“Configuration System Command-Line Arguments” section.

Because of the varying state of the configuration system during initialization processing, use of the `--cinfo` option with the `--init` option may provide incomplete results, or results inconsistent with this manual's description.

--cwduse

Used with option: `--init satk`

Type: CHOICES

The `--cwduse` command-line argument supplies the value for the site.cfg configuration file's SITE section for this configuration option determining how the current working directory is used in conjunction with PATH type options. It supports three choices:

- `first` – the current working directory is always added as the first directory in the search order path regardless of whether it is already present or not
- `last` – the current working directory is added to the directory search order as the last directory if it has has not been specified within the search order, or
- `omit` – directory search order does not automatically contain the current working directory. With this value, the current working directory must explicitly placed in the search order for it to be used.

Default behavior is '`omit`'.

Because current working directory behavior is controlled through the configuration system, when configuration files are disabled (because the user site directory can not be found), search order paths never automatically include the current working directory.

Because options that influence the processing of configuration files must be available first they must reside in the `site.cfg` file.

--force

Used with option: `--init`

Type: EFLAG

The `--force` command-line argument directs the initialization process to overwrite and any existing files. It is available only with the command-line.

--init

Type: CHOICES

The `--init` command-line argument accepts either one of two values:

- `satk`, causing initialization of the `site.cfg` file or

- `site`, causing initialization of the user specified SATK site directory.

input

Used with option: `--tool`

Type: FILENAME

Configuration file option identifying the input file that would be used with the tested tool. If omitted it defaults the the value of the command-line `source` argument.

--satkcfg

Used with option: `--init satk`

Type: DIR

Provides a command-line method for identification of the user's SATK site directory. Alternatively the `SATKCFG` environment variable may be used.

source

Used with option: `--tool`

Type: SOURCE

Command-line positional argument identifying the configuration file section from which the source's options are determined. If a section is not found, only tool and built-in defaults are recognized, including the default value assigned to the input option.

The section identified by the source option will use as the file identified by the `input` configuration file option. The `tstpath` configuration file option and/or `TSTPATH` environment variable identifies the search order path used to locate the file.

--tool

Used with option: `--tool`

Type: CHOICES

Command-line argument identifying the tool whose configuration is being tested. Recognized choices are:

- `config` – this tool.

tstpath

Used with option: `--tool`

Type: PATH

This option allows tests with a search order path. The search order may be specified in a configuration file section as the `tstpath` option or the environment variable `TSTPATH`, or both.