



## SiteCrawl User Manual

### Revision History

Initial Release	16 August 2007
-----------------	----------------

## 1 – General Concepts

### 1.0 - An Important Note

Used inappropriately, programs in the WebTools suite can cause performance problems for servers and networks under analysis. The user is responsible for ensuring they only use these tools to collect information on sites for which they have the appropriate responsibility. The user is advised to examine the documentation thoroughly. The user's email address and name are appended to the User Agent string of all requests made by these tools to facilitate the rapid identification and correction of user errors or mis configurations.

### 1.1 - What is a Web Crawler?

A web crawler is a program designed to obtain information about web sites by navigating them as a human browser might - that is, by following links. By doing so, it can quickly and with minimal initial direction acquire and store information relating to site content, server performance, and efficiency of site implementation. This information can identify possible errors, performance problems, or identify opportunities for optimization – in addition to checking for compliance with site specific policies (for example that every page have a link to Privacy Policy page.)

### 1.1 - What is SiteCrawl?

SiteCrawl is a proprietary web crawler developed by E-Insights, LLC for use by its clients. SiteCrawl includes features that allow search constraints to be quickly and easily specified, and crawl data to be quickly and easily assessed and analyzed. While intended to be reasonably user-friendly, a nominal degree of nuts-and-bolts manipulation is required to use SiteCrawl to its full capacity, as we shall see.

This document describes the use of SiteCrawl in an interactive environment. SiteCrawl can also be configured to perform crawl in batch mode – for example to perform a weekly check on a site as a quality control process.

SiteCrawl is intended for use by Web site owners, or their authorized representatives as a tool to assist in enhancing the quality and performance of their sites. As such, SiteCrawl does not respect “robots.txt” files. EXPLAIN WHY However, SiteCrawl provides a comprehensive range of commands to precisely control what is and is not scanned.

## 1.2 – WebTools & Profiles

SiteCrawl is one of several related tools in the WebTools suite. The manner in which access is gained to these tools is based on the Java Network Launch Protocol (JNLP). Registered users on E-insights, upon logging in to the site using a browser, are provided with links to the tools they are allowed to use. Selecting one of these tools will pass a file with mime type x-application/jnlp back to the browser. If the browser is configured<sup>1</sup> to handle this type then the application will launch. The first time a user starts an E-insights application in this fashion on a specific computer, they will be asked if they wish to trust the code provided by E-insights – an affirmative answer is necessary in order to use the tools. Note that the use of JNLP also ensures that the user always gets access to the most current software – updating is automatic.

These tools also share a common form of authentication, access control and configuration. Normally, once a user has logged into the E-insights web site, then launching a tool as described in the previous paragraph requires no additional authentication. There are rare cases however when the tool itself will ask for authentication. The credentials used are identical to those used on the E-insights.com web site.

Upon launching a WebTools program, you may be required to choose a profile per the pictured dialog box.



Each WebTools-enabled account registered on E-Insights.com has access to one or more profiles. Each profile is attached to different a database in which data generated by WebTools programs is stored and from which said data may be read. A WebTools program launched under a given profile can only read from and write to the database attached to that profile. Otherwise, WebTools programs function identically between profiles; profiles are used solely for information management purposes. Since the specification of the database to use for a specific task is within the Profile, it is possible to have these databases anywhere, including inside corporate firewalls where they are not accessible to (amongst other people) E-insights staff.

## 1.3 - What is a .crl file?

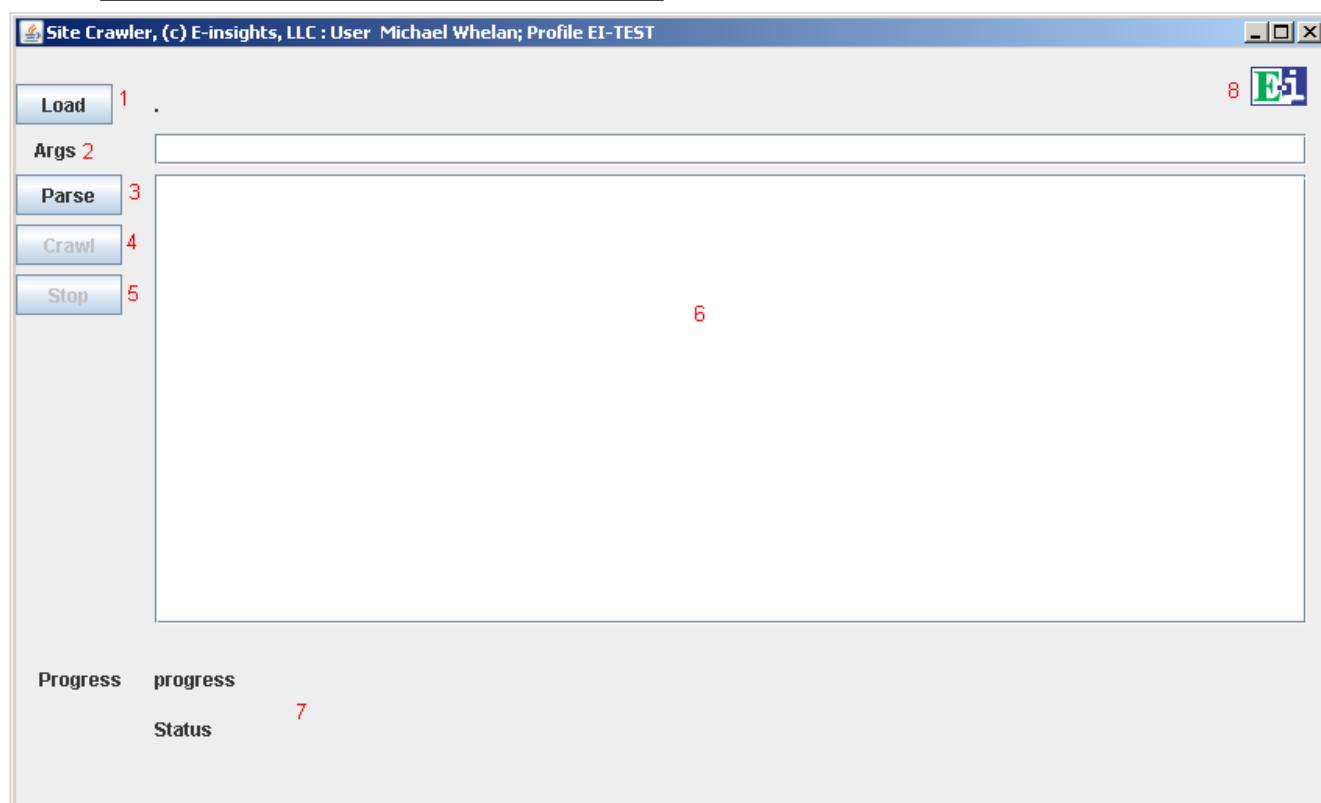
In theory, an unrestricted crawl could continue indefinitely, so long as the pages it read provided the crawler with novel links to follow. In order to specify the range of a crawl,

<sup>1</sup> Many browsers are configured this way by default. If the browser is not so configured, the user will need to download and install the Java Runtime Environment appropriate to their computer from [www.sun.com](http://www.sun.com).

SiteCrawl uses .crl files, which can be created with any basic text editor. .crl files contain lists of rules that SiteCrawl will obey whilst crawling. More information on .crl files and rules can be found below in section 3.

## 2 - Interfaces and Reports

### 2.1 - The SiteCrawl User Interface



**1: The Load Button.** Pressing the load button will open a dialog box prompting the user to select the .crl file containing the rules for this crawl. SiteCrawl requires that a .crl file be loaded before a crawl can commence.

**2: The Arguments Box.** Entering .crl rules into the arguments box will cause them to be appended to the rules specified in the loaded .crl file. Note that doing so has no effect on the .crl file itself; additional arguments are specified for this crawl only. Note that where rules conflict, newer rules override older rules, so the arguments box can be used to fine-tune the rules in a .crl file.

**3: The Parse Button.** Pressing the parse button will cause SiteCrawl to read the rules in the loaded .crl file, as well as those entered via the arguments box. The finished rule set will be displayed in the instruction window (6), and any errors it contains will be indicated. Errors can be corrected by providing a new rule via the arguments box.

**4: The Crawl Button.** Pressing the crawl button will initiate the crawl. The crawl button is grayed out until an acceptable instruction set has been parsed.

5: The Stop button. Pressing the stop button will stop a crawl prematurely, that is, before the conditions specified by the rules have been met. A stopped crawl can not be resumed, but whatever information was obtained before the crawl was stopped will be stored in the appropriate database.

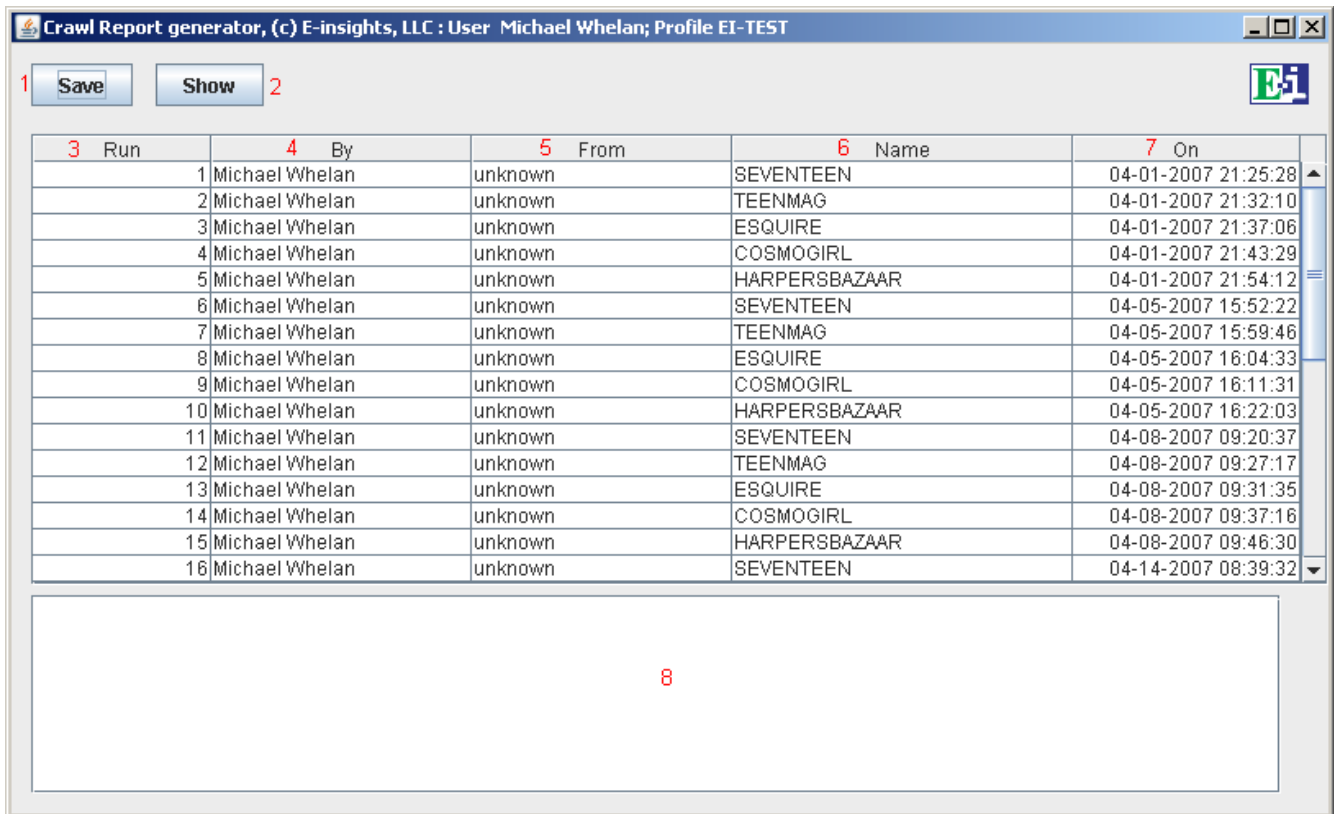
6: The Instruction Window. The instruction window displays the parsed rule set. Any errors contained in the rule set will be indicated here as well.

7: The Progress Indicator. The progress indicator displays information pertaining to a crawl in progress, such as number of pages crawled and estimated percent completed. The progress indicator is updated every 10 seconds, so it may not provide useful information during unusually short crawls.

8: The Logo.

## 2.2 - The Crawl Report Generator

### 2.2.1 - The Report Generator Interface



The screenshot shows a window titled "Crawl Report generator, (c) E-insights, LLC : User Michael Whelan; Profile EI-TEST". At the top left, there are two buttons: "Save" (labeled 1) and "Show" (labeled 2). Below the buttons is a table with the following columns: "Run" (labeled 3), "By" (labeled 4), "From" (labeled 5), "Name" (labeled 6), and "On" (labeled 7). The table contains 16 rows of data. Below the table is a large empty rectangular area (labeled 8) with a red "8" in the center.

3 Run	4 By	5 From	6 Name	7 On
1	Michael Whelan	unknown	SEVENTEEN	04-01-2007 21:25:28
2	Michael Whelan	unknown	TEENMAG	04-01-2007 21:32:10
3	Michael Whelan	unknown	ESQUIRE	04-01-2007 21:37:06
4	Michael Whelan	unknown	COSMOGIRL	04-01-2007 21:43:29
5	Michael Whelan	unknown	HARPERSBAZAAR	04-01-2007 21:54:12
6	Michael Whelan	unknown	SEVENTEEN	04-05-2007 15:52:22
7	Michael Whelan	unknown	TEENMAG	04-05-2007 15:59:46
8	Michael Whelan	unknown	ESQUIRE	04-05-2007 16:04:33
9	Michael Whelan	unknown	COSMOGIRL	04-05-2007 16:11:31
10	Michael Whelan	unknown	HARPERSBAZAAR	04-05-2007 16:22:03
11	Michael Whelan	unknown	SEVENTEEN	04-08-2007 09:20:37
12	Michael Whelan	unknown	TEENMAG	04-08-2007 09:27:17
13	Michael Whelan	unknown	ESQUIRE	04-08-2007 09:31:35
14	Michael Whelan	unknown	COSMOGIRL	04-08-2007 09:37:16
15	Michael Whelan	unknown	HARPERSBAZAAR	04-08-2007 09:46:30
16	Michael Whelan	unknown	SEVENTEEN	04-14-2007 08:39:32

1: The Save Button. Pressing the save button will open a dialog box prompting the user to select a filename and location under which to save a report on the selected crawl.

2: The Show Button. Pressing the show button will display a report on the selected crawl.

Data Set Parameters:

3: The Run Parameter. Each data set is given a run parameter according to the order in which its crawl was conducted. Naturally, run parameters ought to match up with on parameters.

4: The By Parameter. The E-Insights.com username under which the crawl was conducted.

5: The From Parameter. The IP address from which the crawl was conducted.

6: The Name Parameter, per the `testname` rule. See section 3 for more on rules.

7: The On Parameter, or the timestamp. For reference purposes.

8: The Error Box. The error box displays any errors encountered while using the crawl report generator. This box should be empty at all times.

## 2.2.2 - The Crawl Report

The Save and Show buttons on the crawl report generator each produce a crawl report in the form of an HTML file. Crawl reports are designed to be easily readable summaries of crawl data. Each crawl report consists of eight sections, some of which may have additional subsections, depending on the specifics of the crawl.

### **Section 1: Page Summary**

Section 1 presents a table with a row for each unique page crawled, and columns indicating each page's number of elements, errors encountered, and notes (unparseable HTML tags), its size, its loading time, and whether it prompted a redirect. The table can be ordered according to any of these parameters by clicking on a given parameter's column heading.

### **Section 2: External Links**

Section 2 presents a list of URLs linked to by crawled pages that did not satisfy `follow` conditions (see below). Numbered links presented alongside the URLs refer to the page-specific subfield (see 3) of the page on which one of the external links occurred.

Section 3: Individual Pages

Section 3 is divided into subsections, each of which presents detailed information on a single crawled page. Each unique page crawled has one corresponding subsection. Subsections contain a list of the elements referenced on the corresponding page, the URL of each, its size, loading time, error status and file type. Each subsection contains a further sub-subsection listing the external links on the corresponding page.

#### Section 4: Resources

Section 4 presents a list of all embedded objects referenced during the crawl, indicating the error status, loading time, size and URL of each. Mousing over image URLs will cause a thumbnail image to be displayed.

#### Section 5: Hosts

Section 5 is divided into subsections, each of which presents a list of hosts encountered during the crawl. Each root (as specified by the `root` rule; see section 3) has a subsection, and each subsection lists only those hosts encountered during the crawl starting from its associated root.

#### Section 6: Resource Load Errors

Section 6 presents a list of errors encountered while trying to load objects (here called resources) during the crawl. Each is listed with its error type, and the time elapsed between the initial request and the error's receipt.

#### Section 7: Alien Items

Field 7 presents a list of those embedded objects (here called items) encountered during the crawl that were disallowed under an `include` or `exclude` rule (see below), along with their resources and the pages on which they occurred.

#### Section 8: Server Resolution

Field 8 presents a list of the hosts encountered during the crawl, and the resolved IP and server type of each.

## 3 - Rules and Regular Expressions

### 3.1 - .crl Files and Rules

.crl files may be produced in any standard text editor program. Each consists of a number of instructions, one instruction per line. Some particular must be included in order for the crawl to run. The order in which the instructions occur in a .crl file is unimportant, save that where instructions conflict, the latter override the former. What

follows is a list of acceptable instructions:

```
depth
maxpages
testname*
root*
follow*
nofollow
file
timeout
include
exclude
useragent
auth
```

\* indicates that at least one instruction of this type is required by SiteCrawl.

A more detailed account follows:

### **depth**

syntax: `depth=[integer]`

The depth to which links will be followed during the crawl. The root page is depth 1, and a page linked to by a depth *n* page is depth *n*+1. It thus follows that pages can have multiple depth values if they are linked to by other pages with different depth values. No pages will be crawled all of whose depth values exceed that stipulated in the .crl file, except for the root page(s), which are loaded in all cases; consequently, `depth` rule values less than or equal to one will cause the root page(s) to be crawled, but nothing else. The default value for `depth` is 2.

**Example:** `depth=4`

Links occurring on pages of depth 4 will not be followed. Identical links on pages of depth < 4 will still be followed, so long as doing so would not violate any `follow` or `nofollow` instructions.

### **maxpages**

syntax: `maxpages=[integer]`

The maximum total number of pages that will be crawled. The default is 500.

**Example:** `maxpages=750`

SiteCrawl will load no more than 750 pages. Fewer than 750 pages will be loaded if fewer than 750 pages are allowed by `depth`, `follow` and `nofollow` instructions.

### **testname**

syntax: `testname=[string]`

The Name Parameter under which the data produced by the crawl will be stored in the

SiteCrawl database. There is no default value for `testname`; a value must be supplied by the user.

**Example:** `testname="CrawlData"`

SiteCrawl will store the crawl data in the appropriate database under the name `CrawlData`.

### **root**

syntax: `root=[URL]`

The URL of the site at which the crawl begins. There is no default value for `root`; a value must be supplied by the user. Multiple roots can be specified for a single crawl, in which case concurrent crawls will be initiated from each. The page limit stipulated by `maxpages` applies to the aggregate sum of pages loaded; adding additional roots will not cause this limit to be exceeded, unless the number of roots exceeds `maxpages`, in which case the roots will be loaded, but nothing else..

**Example:** `root=http://www.e-insights.com`

SiteCrawl will begin crawling at <http://www.e-insights.com>.

### **follow**

syntax: `follow=[regular expression]`

Stipulates a class of links that SiteCrawl will follow during the course of a crawl. Often, multiple `follow` entries will be necessary to produce useful data. There is no default value for `follow`; a value must be supplied by the user.

**Example:** `follow="^http://www.e-insights.com/.*"`

SiteCrawl will follow all links beginning with "<http://www.e-insights.com/>", unless doing so would violate a `depth` or a `nofollow` instruction. See Appendix II for more information on regular expressions.

### **nofollow**

syntax: `nofollow=[regular expression]`

Stipulates a class of links that SiteCrawl will not follow during the course of a crawl. Useful for limiting the set of possible links allowed by the `follow` instruction. There is no default value for `nofollow`.

**Example:** `nofollow="^http://www.einsights.com/fiddlesticks/.*"`

SiteCrawl will ignore all links beginning with "<http://www.einsights.com/fiddlesticks/>".

**Example:** `nofollow=".*evil.*"`

SiteCrawl will ignore all links containing the string "evil." See Appendix II for more information on regular expressions.

### **file**

syntax: `file=[.crl filename]`

Includes the rules set forth in the indicated `.crl` file. There is no default value for `file`.

**Example:** `file="rules.crl"`

SiteCrawl will insert in-line the rules listed in `rules.crl`. Trying to parse mutually- or self-referencing `.crl` files is a bad idea.

### **timeout**

syntax: `timeout=[real number]`

The longest length of time (in seconds) that SiteCrawl will hold open a connection request. There is no default value for `timeout`; if no value is specified, then SiteCrawl await replies indefinitely. In practice, silent connections will always be broken eventually, but not specifying a `timeout` value can cause the crawl to take a very long time indeed.

**Example:** `timeout=20`

SiteCrawl will wait no longer than 20 seconds for a reply to a request.

### **include**

syntax: `include=[regular expression]`

Stipulates a class of embedded objects that SiteCrawl will attempt to load during the course of a crawl. Objects that do not satisfy the `include` rule will be ignored.

Embedded objects include images and movie files and embedded html documents. Not specifying an `include` rule will cause SiteCrawl to attempt to load all embedded objects.

**Example:** `include="*.png$|*.gif$"`

SiteCrawl will attempt to load only those embedded objects ending with "png" or "gif." Objects that satisfy `include` conditions and `exclude` conditions will not be loaded. See 3.2 for more information of regular expressions.

### **exclude**

syntax: `exclude=[regular expression]`

Stipulates a class of embedded objects that SiteCrawl will not attempt to load during the course of a crawl. Embedded objects include image and movie files and embedded html documents. There is no default value for `exclude`.

**Example:** `exclude="*.eggnog.*"`

SiteCrawl will ignore all embedded objects containing the string "eggnog". See Appendix II for more information of regular expressions.

### **useragent**

syntax: `useragent=[string]`

Specifies the agent name SiteCrawl will include with requests. The agent name specifies browser, operating system, et cetera, and is sometimes relevant to the content a server will provide.

**Example:** useagent="xxx"

---

### **auth**

syntax: auth=[string:string]

Specifies the username and password SiteCrawl will supply if prompted to do so during the course of a crawl. There is no default value for auth. SiteCrawl currently supports HTTP Basic Authentication to a single site only.

**Example:** auth="gyges:briareus"

SiteCrawl will use "gyges" as a username and "briareus" as a password if so prompted.

## Appendix I - Regular Expressions

Several of SiteCrawl's instructions require the use of regular expressions, whereby a large class of acceptable string values may be succinctly described. SiteCrawl uses the "Java Regex" class for handling regular expressions, a full description of which can be found at <http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/package-summary.html> .

The basic syntax is as follows:

Functional (or special) characters are those that have special meaning within the context of regular expressions. These are all characters `".*?|()[]-{}^$\"`

**x**, where x is a non-functional character: the character x

**\x**, where x is a functional character: the character x

**x\***, where x is a non-functional character: any number of successive xs

**.**: any single character

**^**: the beginning of the string.

**\$**: the end of the string.

**x|y**, where x and y are strings: x or y.

**[x-y]**, where x and y are characters, and the ASCII value of y is greater than or equal to that of x: any single character whose ASCII value falls between that of x and that of y, inclusively.

**z{x,y}**, where x and y are integers, and y >= x, and z is a string: any number of successive zs in the position of z, so long as there are no fewer than x and no more than y.

**(x)y**, where x is a string and y is a functional character: applies y to x as though x were a character.

Note that unless otherwise specified with a **^** or **\$** character, all SiteCrawl regular expressions are assumed to begin and end with **.**. Nevertheless, for the sake of clarity, it is often sensible to explicitly include **.**, where appropriate.

### Some Examples:

**.\*awl\$**

Accepts all strings ending with "awl". "Crawl", "brawl", "shawl", "nHy8d^^4wawl" are all acceptable; "trawling" is not.

**^[0-9][0-9]\*\$**

Accepts all strings that consist of one decimal digit, followed by arbitrarily many decimal digits. Equivalently, accepts all integers (including those with leading zeros), including zero, but excluding the empty string.

**`^00*[[1-9][0-9]*$`**

Accepts integers including zero (and any number of successive zeros), but excluding non-zero integers with leading zeros and empty strings.

**`^e-insights\.com$`**

Accepts “e-insights.com” and no other strings.

## **Appendix II – Examples**

MW check does # have to be first char ??

depth=3

maxpages=100

root=<http://www.e-insights.com>

# dont give any include statement – defaults to include everything

exclude="\*.rss\$" # dont load syndication files

exclude="\*.mov\$" # dont load movies

follow="^<http://www.e-insights.com/>.\*" # follow sites www links

follow="^<http://e-insights.com/>.\*" # and without the www

nofollow="^[http://www.e-insights.com/site\\_search/](http://www.e-insights.com/site_search/).\*" # dont follow search links

testname="Simple Crawl" # give it a name