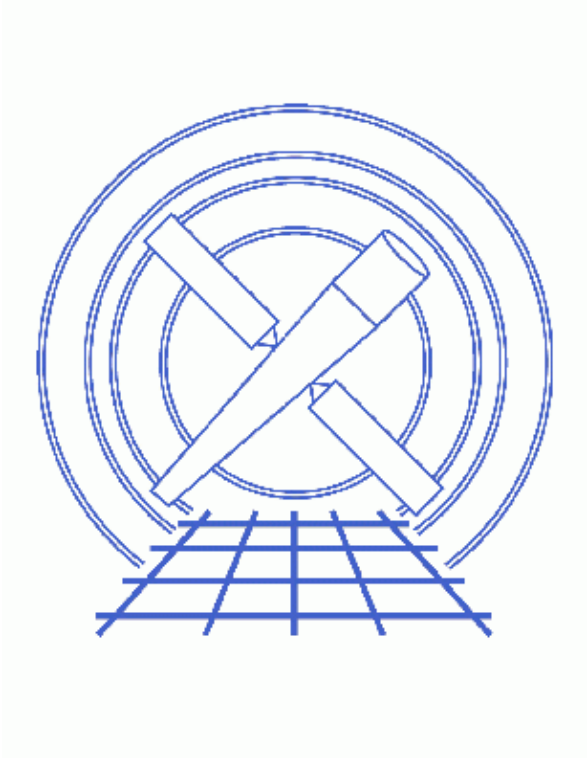
Sherpa and Scripts



Sherpa Threads (CIAO 3.4)

Table of Contents

URL: http://cxc.harvard.edu/sherpa/threads/scripts/ Last modified: 1 Dec 2006

Sherpa and Scripts

Sherpa Threads

Overview

Last Update: 1 Dec 2006 − updated for CIAO 3.4: *Sherpa* version

Synopsis:

In addition to accepting interactive input, *Sherpa* can read commands from a script file. This thread discusses how to drive *Sherpa* using both native *Sherpa* scripts and S−Lang scripts.

Related Links:

- Customizing Sherpa with a Resource File
- A Guide to the S−Lang Language

Proceed to the HTML or hardcopy (PDF: A4 | letter) version of the thread.

Sherpa Scripts

A *Sherpa* script is simply a text file that contains *Sherpa* commands. Anything that can be entered at the *Sherpa* prompt may also appear in a script. Hence, in addition to *Sherpa* commands, a script may contain *ChIPS* commands and single−line S−Lang statements. Within a script file, any line that begins with a # character is interpreted as a comment and not evaluated. Empty lines are also ignored.

The example script below contains three comment lines, two *Sherpa* commands (`PARAMPROMPT` and `SOURCE`), one *ChIPS* command (`CLEAR`), and two S−Lang statements (`message(...)` and `list_par`):

```
unix% more script1.shp
# Sherpa commands
paramprompt off
source = powlaw1d[src1]

# ChIPS command
clear

# S-Lang statements
message("Current parameter values:")
list_par
```

You may execute this script during a *Sherpa* session via the `USE` command:

```
sherpa> use script1.shp
Model parameter prompting is off
Current parameter values:
  #       Name Type       Value Lnk Frz         Min         Max       Delta
```

```
  1 src1.gamma  src           1   0   0         −10          10          −1
  2   src1.ref  src           1   0   1     −1e+120      1e+120          −1
  3  src1.ampl  src           1   0   0           0      1e+120          −1
sherpa>
```

After the script runs, the *Sherpa* prompt reappears, allowing you to continue your session.

You may also run the script at the start of your session by supplying the script name as a command−line argument to *Sherpa*:

```
unix% sherpa script1.shp

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Welcome to Sherpa: CXC's Modeling and Fitting Program
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Version: CIAO 3.4

Type AHELP SHERPA for overview.
Type EXIT, QUIT, or BYE to leave the program.

Notes:
    Temporary files for visualization will be written to the directory:
    /tmp
    To change this so that these files are not deleted when you exit Sherpa,
    edit $ASCDS_WORK_PATH in your 'ciao' setup script.

    Abundances set to Anders & Grevesse
Model parameter prompting is off
Current parameter values:
  #       Name Type       Value Lnk Frz         Min         Max       Delta
  1 src1.gamma  src           1   0   0         −10          10          −1
  2   src1.ref  src           1   0   1     −1e+120      1e+120          −1
  3  src1.ampl  src           1   0   0           0      1e+120          −1

sherpa>
```

Note that if a *Sherpa* resource file exists, *Sherpa* will load it *before* any script specified on the command line. This apples to both *Sherpa* scripts and S−Lang scripts. (See the thread Customizing Sherpa with a Resource File for more information on using resource files.)

Finally, if you wish to run *only* the script (and not enter interactive mode), you can specify the `−−batch` option before the script name when starting *Sherpa*:

```
unix% sherpa −−batch script1.shp
    Abundances set to Anders & Grevesse
Model parameter prompting is off
Current parameter values:
  #       Name Type       Value Lnk Frz         Min         Max       Delta
  1 src1.gamma  src           1   0   0         −10          10          −1
  2   src1.ref  src           1   0   1     −1e+120      1e+120          −1
  3  src1.ampl  src           1   0   0           0      1e+120          −1
unix%
```

In this case, *Sherpa* exits as soon as the script completes, so the prompt never appears.

S−Lang Scripts

A standard *Sherpa* script can contain only single−line S−Lang statements. However, it is also possible to run S−Lang scripts from within *Sherpa*. Such scripts may contain any valid S−Lang code, including function definitions and multi−line statements. Note that within a S−Lang script, you must declare variables before

using them (e.g. "`variable foo;`") and end each statement with a semi−colon.

The following example S−Lang script checks whether a source model expression is currently defined (using the `get_source_expr` function). If no source expression is found, it sets one; otherwise, it issues a message saying that a source expression already exists.

```
unix% more script2.sl
if (get_source_expr() == NULL) {
    () = set_source_expr("powlaw1d[src2]");
    message("Set new source expression");
} else {
    message("Source expression already defined");
}
```

You can run this script from within *Sherpa* by using the `evalfile` function, which takes the name of a script as its argument and executes the script:

```
sherpa> () = evalfile("script2.sl")
Set new source expression
sherpa> () = evalfile("script2.sl")
Source expression already defined
sherpa> show source
Source 1: src2
powlaw[src2]  (integrate: on)
    Param   Type      Value        Min        Max                Units
    -----   ----      -----        ---        ---                -----
 1  gamma thawed          1        -10         10
 2    ref frozen          1    -1e+120     1e+120
 3   ampl thawed          1          0     1e+120
sherpa>
```

You may also run the script at *Sherpa* startup by specifying `--slscript` and the script name as command−line arguments to *Sherpa*:

```
unix% sherpa --slscript script2.sl
...
    Abundances set to Anders & Grevesse
Set new source expression

sherpa> show source
Source 1: src2
powlaw[src2]  (integrate: on)
    Param   Type      Value        Min        Max                Units
    -----   ----      -----        ---        ---                -----
 1  gamma thawed          1        -10         10
 2    ref frozen          1    -1e+120     1e+120
 3   ampl thawed          1          0     1e+120
sherpa>
```

As with standard *Sherpa* scripts, you can tell *Sherpa* to exit after running the script by adding `--batch` before `--slscript`. You may also run more than one S−Lang script from the command line; `--slscript` must precede the name of each script.

Finally, it is also possible to run both S−Lang scripts *and* standard *Sherpa* scripts at startup:

```
unix% sherpa --slscript script2.sl script1.shp
```

Multiple scripts of both types may be run via a single command line. However, all S−Lang scripts must be listed before any *Sherpa* script:

```
unix% sherpa --slscript 1.sl --slscript 2.sl a.shp b.shp
```

Using Sherpa Functions Outside of Sherpa

The Sherpa/S−Lang module allows one to employ the full functionality of *Sherpa* without invoking the `sherpa` executable at all. The command `import("sherpa")` makes the *Sherpa* module available to any S−Lang script or S−Lang−enabled application. For example, one may import *Sherpa* into *ChIPS*:

```
chips> import("sherpa")
    Abundances set to Anders & Grevesse
chips> get_method_expr
levenberg−marquardt
chips> () = sherpa_eval("show statistic")
Statistic:           Chi−Squared Gehrels
```

(Note that the `sherpa_eval` function can be very useful in this context. It takes a string as its argument and interprets the string as a *Sherpa* command entered at the *Sherpa* prompt. This allows an application or script that imports the *Sherpa* module to execute *any Sherpa* command. However, `sherpa_eval` differs from the actual *Sherpa* command line in that one may execute only *Sherpa* commands, not *ChIPS* commands or S−Lang statements. To execute a *ChIPS* command in a S−Lang script, use `chips_eval`.)

The *Sherpa* module may also be imported into an `slsh` script, which allows one to write standalone, command−line scripts that use *Sherpa*. For example, the script `show_model_defaults` takes the name of a *Sherpa* model as its argument and displays the default parameter values for that model:

```
unix% more show_model_defaults
#!/usr/bin/env slsh

import("sherpa");

variable model = __argv[1];

!if (create_model(model)) {
    message("Cannot determine defaults for model " + model);
} else {
    message("Parameter defaults for model " + model + ": ");
    list_par();
}
```

You can use the script as follows:

```
unix% chmod +x show_model_defaults
unix% ./show_model_defaults foo
    Abundances set to Anders & Grevesse
Cannot determine defaults for model foo
unix% ./show_model_defaults gauss
    Abundances set to Anders & Grevesse
Parameter defaults for model gauss:
  #       Name Type       Value Lnk Frz          Min          Max        Delta
  1 gauss.fwhm  src          10   0   0       1e−120       1e+120           −1
  2  gauss.pos  src           0   0   0      −1e+120       1e+120           −1
  3 gauss.ampl  src           1   0   0      −1e+120       1e+120           −1
```

A Note on Script Names

In this thread, we have identified *Sherpa* scripts with a `.shp` extension and S−Lang scripts with a `.sl` extension. While this is a useful convention, it is *not* a requirement. You are free to name your scripts as you see fit.

History

14 Jan 2005	updated for CIAO 3.2: minor changes to screen output
21 Dec 2005	reviewed for CIAO 3.3: no changes
01 Dec 2006	updated for CIAO 3.4: *Sherpa* version

URL: http://cxc.harvard.edu/sherpa/threads/scripts/ Last modified: 1 Dec 2006