Title

stata.com

translate — Print and translate logs

Syntax

Print log and SMCL files

```
print filename [ , like(ext) name(windowname) override_options ]
```

Translate log files to SMCL files and vice versa

```
translate filename_in filename_out [ , translator(tname) name(windowname)
    override_options replace ]
```

View translator parameter settings

```
translator query [tname]
```

Change translator parameter settings

```
translator set [tname setopt setval]
```

Return translator parameter settings to default values

```
translator reset tname
```

List current mappings from one extension to another

```
transmap query [ .ext ]
```

Specify that files with one extension be treated the same as files with another extension

```
transmap define .ext_new .ext_old
```

filename in `print`, in addition to being a filename to be printed, may be specified as `@Results` to mean the Results window and `@Viewer` to mean the Viewer window.

filename$_{\text{in}}$ in `translate` may be specified just as *filename* in `print`.

tname in `translator` specifies the name of a translator; see the `translator()` option under *Options for translate*.

Description

`print` prints log, SMCL, and text files. Although there is considerable flexibility in how `print` (and `translate`, which `print` uses) can be set to work, they have already been set up and should just work:

```
. print mylog.smcl
. print mylog.log
```

Unix users may discover that they need to do a bit of setup before `print` works; see *Printing files, Unix* below. International Unix users may also wish to modify the default paper size. All users can tailor `print` and `translate` to their needs.

`print` may also be used to print the current contents of the Results window or the Viewer. For instance, the current contents of the Results window could be printed by typing

```
. print @Results
```

`translate` translates log and SMCL files from one format to another, the other typically being suitable for printing. `translate` can also translate SMCL logs (logs created by typing, say, `log using mylog`) to plain text:

```
. translate mylog.smcl mylog.log
```

You can use `translate` to recover a log when you have forgotten to start one. You may type

```
. translate @Results mylog.txt
```

to capture as plain text what is currently shown in the Results window.

This entry provides a general overview of `print` and `translate` and covers in detail the printing and translation of text (nongraphic) files.

`translator query`, `translator set`, and `translator reset` show, change, and restore the default values of the settings for each translator.

`transmap define` and `transmap query` create and show mappings from one file extension to another for use with `print` and `translate`.

For example, `print myfile.txt` knows to use a translator appropriate for printing text files because of the `.txt` extension. However, it does not know what to do with `.xyz` files. If you have `.xyz` files and always wish to treat them as `.txt` files, you can type `transmap define .xyz .txt`.

Options for print

`like(`*ext*`)` specifies how the file should be translated to a form suitable for printing. The default is to determine the translation method from the extension of *filename*. Thus `mylog.smcl` is translated according to the rule for translating `smcl` files, `myfile.txt` is translated according to the rule for translating `txt` files, and so on. (These rules are, in fact, `translate`'s `smcl2prn` and `txt2prn` translators, but put that aside for the moment.)

Rules for the following extensions are predefined:

`.txt`	assume input file contains plain text
`.log`	assume input file contains Stata log text
`.smcl`	assume input file contains SMCL

To print a file that has an extension different from those listed above, you can define a new extension, but you do not have to do that. Assume that you wish to print the file `read.me`, which you know to contain plain text. If you were just to type `print read.me`, you would be told that Stata cannot translate `.me` files. (You would actually be told that the translator for `me2prn` was not found.) You could type `print read.me, like(txt)` to tell `print` to print `read.me` like a `.txt` file.

On the other hand, you could type

```
. transmap define .me .txt
```

to tell Stata that `.me` files are always to be treated like `.txt` files. If you did that, Stata would remember the new rule, even in future sessions.

When you specify the `like()` option, you override the recorded rules. So, if you were to type `print mylog.smcl, like(txt)`, the file would be printed as plain text (meaning that all the SMCL commands would show).

`name(`*windowname*`)` specifies which window to print when printing a Viewer. The default is for Stata to print the topmost Viewer [Unix(GUI) users: See the second technical note in *Printing files, Unix*]. The `name()` option is ignored when printing the Results window.

The window name is located inside parentheses in the window title. For example, if the title for a Viewer window is *Viewer (#1) [help print]*, the name for the window is `#1`.

override_options refers to `translate`'s options for overriding default values. `print` uses `translate` to translate the file into a format suitable for sending to the printer, and thus `translate`'s *override_options* may also be used with `print`. The settings available vary between each translator (for example, `smcl2ps` will have different settings than `smcl2txt`) and may also differ across operating systems (for example, Windows may have different printing options than Mac OS X). To find out what you can override when printing `.smcl` files, type

```
. translator query smcl2prn
(output omitted)
```

In the omitted output, you might learn that there is an `rmargin` # tunable value, which specifies the right margin in inches. You could specify the *override_option* `rmargin(#)` to temporarily override the default value, or you could type `translator set smcl2prn rmargin` # beforehand to permanently reset the value.

Alternatively, on some computers with some translators, you might discover that nothing can be set.

Options for translate

`translator(`*tname*`)` specifies the name of the translator to be used to translate the file. The available translators are

tname	Input	Output
`smcl2ps`	SMCL	PostScript
`log2ps`	Stata text log	PostScript
`txt2ps`	generic text file	PostScript
`Viewer2ps`	Viewer window	PostScript
`Results2ps`	Results window	PostScript
`smcl2prn`	SMCL	default printer format
`log2prn`	Stata text log	default printer format
`txt2prn`	generic text log	default printer format
`Results2prn`	Results window	default printer format
`Viewer2prn`	Viewer window	default printer format
`smcl2txt`	SMCL	generic text file
`smcl2log`	SMCL	Stata text log
`Results2txt`	Results window	generic text file
`Viewer2txt`	Viewer window	generic text file
`smcl2pdf`	SMCL	PDF
`log2pdf`	Stata text log	PDF
`txt2pdf`	generic text log	PDF
`Results2pdf`	Results window	PDF
`Viewer2pdf`	Viewer window	PDF

If `translator()` is not specified, `translate` determines which translator to use from extensions of the filenames specified. Typing `translate myfile.smcl myfile.ps` would use the `smcl2ps` translator. Typing `translate myfile.smcl myfile.ps, translate(smcl2prn)` would override the default and use the `smcl2prn` translator.

Actually, when you type `translate` *a*`.`*b* *c*`.`*d*, `translate` looks up `.`*b* in the `transmap` extension-synonym table. If `.`*b* is not found, the translator *b*`2`*d* is used. If `.`*b* is found in the table, the mapped extension is used (call it *b*′), and then the translator *b*′`2`*d* is used. For example,

Command	Translator used
`. translate myfile.smcl myfile.ps`	`smcl2ps`
`. translate myfile.odd myfile.ps`	`odd2ps`, which does not exist, so error
`. transmap define .odd .txt`	
`. translate myfile.odd myfile.ps`	`txt2ps`

You can list the mappings that `translate` uses by typing `transmap query`.

`name(`*windowname*`)` specifies which window to translate when translating a Viewer. The default is for Stata to translate the topmost Viewer. The `name()` option is ignored when translating the Results window.

The window name is located inside parentheses in the window title. For example, if the title for a Viewer window is *Viewer (#1) [help print]*, the name for the window is `#1`.

override_options override any of the default options of the specified or implied translator. To find out what you can override for, say, `log2ps`, type

```
. translator query log2ps
(output omitted)
```

In the omitted output, you might learn that there is an `rmargin` # tunable value, which, for `log2ps`, specifies the right margin in inches. You could specify the *override_option* `rmargin(#)` to temporarily override the default value or type `translator set log2ps rmargin` # beforehand to permanently reset the value.

`replace` specifies that *filename*$_{\text{out}}$ be replaced if it already exists.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Printing files

Printing should be easy; just type

```
. print mylog.smcl
. print mylog.log
```

You can use `print` to print SMCL files, plain text files, and even the contents of the Results and Viewer windows:

```
. print @Results
. print @Viewer
. print @Viewer, name(#2)
```

For information about printing and translating graph files, see [G-2] **graph print** and see [G-2] **graph export**.

Printing files, Mac and Windows

When you type `print`, you are using the same facility that you would be using if you had selected **Print** from the **File** menu. If you try to print a file that Stata does not know about, Stata will complain:

```
. print read.me
translator me2prn not found
(perhaps you need to specify the like() option)
r(111);
```

Then you could type

```
. print read.me, like(txt)
```

to indicate that you wanted `read.me` sent to the printer in the same fashion as if the file were named `readme.txt`, or you could type

```
. transmap define .me .txt
. print read.me
```

Here you are telling Stata once and for all that you want files ending in `.me` to be treated in the same way as files ending in `.txt`. Stata will remember this mapping, even across sessions. To clear the `.me` mapping, type

```
. transmap define .me
```

To see all the mappings, type

```
. transmap query
```

To print to a file, use the `translate` command, not `print`:

```
. translate mylog.smcl mylog.prn
```

`translate` prints to a file by using the Windows print driver when the new filename ends in `.prn`. Under Mac, the `prn` translators are the same as the `pdf` translators. We suggest that you simply use the `.pdf` file extension when printing to a file.

Printing files, Unix

Stata assumes that you have a PostScript printer attached to your Unix computer and that the Unix command `lpr(1)` can be used to send PostScript files to it, but you can change this. On your Unix system, typing

```
mycomputer$ lpr < filename
```

may not be sufficient to print PostScript files. For instance, perhaps on your system you would need to type

```
mycomputer$ lpr -Plexmark < filename
```

or

```
mycomputer$ lpr -Plexmark filename
```

or something else. To set the print command to be `lpr -Plexmark` *filename* and to state that the printer expects to receive PostScript files, type

```
. printer define prn ps "lpr -Plexmark @"
```

To set the print command to `lpr -Plexmark <` *filename* and to state that the printer expects to receive plain text files, type

```
. printer define prn txt "lpr -Plexmark < @"
```

That is, just type the command necessary to send files to your printer and include an `@` sign where the filename should be substituted. Two file formats are available: `ps` and `txt`. The default setting, as shipped from the factory, is

```
. printer define prn ps "lpr < @"
```

We will return to the `printer` command in the technical note that follows because it has some other capabilities you should know about.

In any case, after you redefine the default printer, the following should just work:

```
. print mylog.smcl
. print mylog.log
```

If you try to print a file that Stata does not know about, it will complain:

```
. print read.me
translator me2prn not found
r(111);
```

Here you could type

```
. print read.me, like(txt)
```

to indicate that you wanted `read.me` sent to the printer in the same fashion as if the file were named `readme.txt`, or you could type

```
. transmap define .me .txt
. print read.me
```

Here you are telling Stata once and for all that you want files ending in `.me` to be treated in the same way as files ending in `.txt`. Stata will remember this setting for `.me`, even across sessions.

If you want to clear the `.me` setting, type

```
. transmap define .me
```

If you want to see all your settings, type

```
. transmap query
```

❑ Technical note

The syntax of the `printer` command is

`printer define` *printername* [{ `ps` | `txt` } "*Unix command with* `@`"]

`printer query` [*printername*]

You may define multiple printers. By default, `print` uses the printer named `prn`, but `print` has the syntax

`print` *filename* [`, like(`*ext*`) printer(`*printername*`)` *override_options*]

so, if you define multiple printers, you may route your output to them.

For instance, if you have a second printer on your system, you might type

```
. printer define lexmark ps "lpr -Plexmark < @"
```

After doing that, you could type

```
. print myfile.smcl, printer(lexmark)
```

Any printers that you set will be remembered even across sessions. You can delete printers:

```
. printer define lexmark
```

You can list all the defined printers by typing `printer query`, and you can list the definition of a particular printer, say, `prn`, by typing `printer query prn`.

The default printer `prn` we have predefined for you is

```
. printer define prn ps "lpr < @"
```

meaning that we assume that it is a PostScript printer and that the Unix command `lpr(1)`, without options, is sufficient to cause files to print. Feel free to change the default definition. If you change it, the change will be remembered across sessions. ❑

❑ Technical note

Unix(GUI) users should note that X-Windows does not have the concept of a window z-order, which prevents Stata from determining which window is the topmost window. Instead, Stata determines which window is topmost based on which window has the focus. However, some window managers will set the focus to a window without bringing the window to the top. What Stata considers the topmost window may not appear topmost visually. For this reason, you should always use the `name()` option to ensure that the correct window is printed. ❑

❑ Technical note

When you select the Results window to print from the **Print** menu or toolbar button, the result is the same as if you were to issue the `print` command. When you select a Viewer window to print from the **Print** menu or toolbar button, the result is the same as if you were to issue the `print` command with a `name()` option. ❑

The translation to PostScript format is done by `translate` and, in particular, is performed by the translators `smcl2ps`, `log2ps`, and `txt2ps`. There are many tunable parameters in each of these translators. You can display the current values of these tunable parameters for, say, `smcl2ps` by typing

```
. translator query smcl2ps
(output omitted)
```

and you can set any of the tunable parameters (for instance, setting `smcl2ps`'s `rmargin` value to 1) by typing

```
. translator set smcl2ps rmargin 1
(output omitted)
```

Any settings you make will be remembered across sessions. You can reset `smcl2ps` to be as it was when Stata was shipped by typing

```
. translator reset smcl2ps
```

Translating files from one format to another

If you have a SMCL log, which you might have created by previously typing `log using mylog`, you can translate it to an text log by typing

```
. translate myfile.smcl myfile.log
```

and you can translate it to a PostScript file by typing

```
. translate myfile.smcl myfile.ps
```

`translate` translates files from one format to another, and, in fact, `print` uses `translate` to produce a file suitable for sending to the printer.

When you type

```
. translate a.b c.d
```

`translate` looks for the predefined translator *b*2*d* and uses that to perform the translation. If there is a `transmap` synonym for *b*, however, the mapped value *b*′ is used: *b*′2*d*.

Only certain translators exist, and they are listed under the description of the `translate()` option in *Options for translate* above, or you can type

```
. translator query
```

for a complete (and perhaps more up-to-date) list.

Anyway, `translate` forms the name *b*2*d* or *b*′2*d*, and if the translator does not exist, `translate` issues an error message. With the `translator()` option, you can specify exactly which translator to use, and then it does not matter how your files are named.

The only other thing to know is that some translators have tunable parameters that affect how they perform their translation. You can type

```
. translator query translator_name
```

to find out what those parameters are. Some translators have no tunable parameters, and some have many:

```
. translator query smcl2ps
```

header	on						
headertext							
logo	on						
user							
projecttext							
cmdnumber	on						
fontsize	9			lmargin	1.00		
pagesize	letter			rmargin	1.00		
pagewidth	8.50			tmargin	1.00		
pageheight	11.00			bmargin	1.00		
scheme	monochrome						
cust1_result_color	0	0	0	cust2_result_color	0	0	0
cust1_standard_color	0	0	0	cust2_standard_color	0	0	0
cust1_error_color	0	0	0	cust2_error_color	255	0	0
cust1_input_color	0	0	0	cust2_input_color	0	0	0
cust1_link_color	0	0	0	cust2_link_color	0	0	255
cust1_hilite_color	0	0	0	cust2_hilite_color	0	0	0
cust1_result_bold	on			cust2_result_bold	on		
cust1_standard_bold	off			cust2_standard_bold	off		
cust1_error_bold	on			cust2_error_bold	on		
cust1_input_bold	off			cust2_input_bold	off		
cust1_link_bold	off			cust2_link_bold	off		
cust1_hilite_bold	on			cust2_hilite_bold	on		
cust1_link_underline	on			cust2_link_underline	on		
cust1_hilite_underline	off			cust2_hilite_underline	off		

You can temporarily override any setting by specifying the *setopt*(*setval*) option on the `translate` (or `print`) command. For instance, you can type

```
. translate ..., ... cmdnumber(off)
```

or you can reset the value permanently by typing

```
. translator set smcl2ps setopt setval
```

For instance,

```
. translator set smcl2ps cmdnumber off
```

If you reset a value, Stata will remember the change, even in future sessions.

Mac and Windows users: The `smcl2ps` (and the other `*2ps` translators) are not used by `print`, even when you have a PostScript printer attached to your computer. Instead, the Mac or Windows print driver is used. Resetting `smcl2ps` values will not affect printing; instead, you change the defaults in the Printers Control Panel in Windows and by selecting **Page Setup...** from the **File** menu in Mac. You can, however, `translate` files yourself using the `smcl2ps` translator and the other `*2ps` translators.

Stored results

`transmap query` *.ext* stores in macro `r(suffix)` the mapped extension (without the leading period) or stores *ext* if the *ext* is not mapped.

`translator query` *translatorname* stores *setval* in macro `r(`*setopt*`)` for every *setopt*, *setval* pair.

`printer query` *printername* (Unix only) stores in macro `r(suffix)` the "filetype" of the input that the printer expects (currently "`ps`" or "`txt`") and, in macro `r(command)`, the command to send output to the printer.

Also see

[R] **log** — Echo copy of session to file

[G-2] **graph export** — Export current graph

[G-2] **graph print** — Print a graph

[G-2] **graph set** — Set graphics options

[P] **smcl** — Stata Markup and Control Language

[U] 15 Saving and printing output—log files