

# The `bibexport.sh` script

Nicolas Markey

2005/02/27

## Abstract

`bibexport.sh` is a small shell script, relying on Bib<sub>T</sub>E<sub>X</sub>, that extracts entries of one or several `.bib` file(s). It will expand abbreviations and cross-references, except standard month and journal abbreviations. The output is indented as neatly as possible, yielding a readable `.bib` file even if the original one was not.

## 1 Exporting `.bib` files

### 1.1 Why and how?

Bib<sub>T</sub>E<sub>X</sub> aims at allowing for the use of one single `.bib` file, containing many entries, from which Bib<sub>T</sub>E<sub>X</sub> extracts only the `\cited` ones. When sending a document to someone else, this requires either to send the whole file, or to extract some entries from the `.bib` file.

Bib<sub>T</sub>E<sub>X</sub> also has a mechanism for using abbreviations and cross-references. When extracting entries of a large `.bib` file, it can be interesting to develop those abbreviations, in order to get a clean, self-contained `.bib` file. Also, it may be useful to develop cross-references in a `.bib` file, independently of any document.

`bibexport` can either extract entries that are cited in a document, or all the entries of one or several `.bib` files. It will always develop cross-references and abbreviations, except standard abbreviations for months or some journals, that are defined in standard Bib<sub>T</sub>E<sub>X</sub> styles. This script uses Bib<sub>T</sub>E<sub>X</sub>. This has both pros and cons:

- + it is very simple. Basically, the script simply calls Bib<sub>T</sub>E<sub>X</sub>, and the `.bst` file just outputs the name and the content of each field.
- + since it uses Bib<sub>T</sub>E<sub>X</sub>, we are sure that it will handle everything "properly", *i.e.* in the same way as they will be handled when cited in a L<sup>A</sup>T<sub>E</sub>X document;
- = Bib<sub>T</sub>E<sub>X</sub> has some strict limitations (especially "no more than 78 consecutive non-space characters") that we must be aware of. On the other hand, any such problem occurring within the script would also occur when compiling a document;
- abbreviations and cross-references will *always* be developed. It could be argued that this is also a positive point, but having the choice would have been better.

- Many people seem to find BibTeX's internal language clumsy, and thus the script could be difficult to adapt to special needs. However, this is *really* not that difficult, as will be explained later on. And please, don't hesitate to ask for any improvement or for the addition of some extra (but still relevant) fields.

## 1.2 Related scripts

Several other tools exist for achieving this task:

- `aux2bib`, written by Ralf Treinen, relies on `bib2bib`, which is a CAML program for selecting some entries in one or several `.bib` files. It does not expand anything, but includes all the necessary definitions and entries.
- `bibextract.sh`, by Nelson Beebe. This script uses AWK for extracting some entries out of a `.bib` file. It is said to be noncompliant with cross-references.
- `subset.bst`, by David Kotz. `export.bst` develops the same ideas (but I discovered that only later on). `subset.bst` does not handle `@preamble`, neither does it "protect" standard abbreviations.
- Probalby some others...

## 1.3 Some examples

- extracting `\cited` references of a document, also including cross-references:

```
bibexport -o <result>.bib <file>.aux
```

- extracting `\cited` references of a document, without crossrefs, and using a special `.bst` file:

```
bibexport -b <style>.bst -o <result>.bib <file>.aux
```

- export all the entries of two `.bib` files (including crossrefed entries):

```
bibexport -a -o <result>.bib <file1>.bib <file2>.bib
```

- export all the entries of two `.bib` files (without crossrefs):

```
bibexport -a -n -o <result>.bib <file1>.bib <file2>.bib
```

In fact, the only difference between this and the previous one is that `crossref` field will be filtered out at the end of the script.

- export all the entries of two `.bib` files, using an extra file containing cross-referenced entries (that should not be included):

```
bibexport -a -e <crossref>.bib -n -o <result>.bib \
    <file1>.bib <file2>.bib
```

## 2 The code

### 2.1 The shell script

#### 2.1.1 Initialization

usage() We first define how the script should be used:

```
1 (*script)
2 function usage()
3 {
4 echo "bibexport: a tool to extract BibTeX entries out of .bib files.
5 usage: $0 [-h|v] [-n] [-b bst] [-a [-e file]...] [-o file] file...
6 -a, --all          export the entire .bib files
7 -b, --bst          specifies the .bst style file [default: export.bst]
8 -e, --extra        extra .bib files to be used (for crossrefs)
9 -c, --crossref    include entries that are crossref'd [default: yes]
10 -n, --no-crossref don't include crossref'd entries [default: no]
11 -o file           write output to file [default: bibexport.bib]
12 -h, --help       print this message and exit
13 -v, --version    print version number and exit";
14 exit 0;
15 }
16 (/script)
```

CREF We define the default value of some variables:  
FILE  
EXT  
EXTRA  
EXTRABIB  
SPACE  
BST

- \$CREF: the value of `-min-crossrefs`,
- \$FILE: the input file(s),
- \$EXT: the extension (`.aux` or `.bib`) of input files,
- \$EXTRA: list of possible extra `.bib` files without extension,
- \$EXTRABIB: list of possible extra `.bib` files with extension,
- \$SPACE: file name separator (can be `_`, comma or empty),
- \$BST: the `.bst` file to be used.

```
17 (*script)
18 CROSSREF="1";
19 FILE="";
20 EXT=".aux";
21 EXTRA="";
22 EXTRABIB="";
23 SPACE=" ";
24 BST="export";
25 (/script)
```

#### 2.1.2 Handling arguments

If no argument have been supplied, we call `usage()`.

```
26 (*script)
27 if [ $# -eq 0 ]; then
28 usage;
```

```
29 fi
30 </script>
```

Otherwise, we enter a `while`-loop for handling the whole list of arguments:

```
31 (*script)
32 while [ $# != 0 ]; do
33     case $1 in
34 </script>
```

- `-a` or `--all`: export all the bibliography. This means that we input `.bib` files.

```
35     (*script)
36         -a|--all)
37         EXT=""; SPACE="";
38         shift ;;
39     </script>
```

- `-b` or `--bst`: specifies the style file. It seems that Bib<sub>T</sub>E<sub>X</sub> does not like the `./style.bst` syntax, and we have to handle that case separately.

```
40     (*script)
41         -b|--bst)
42         if [ "'dirname $2'" = "." ]; then
43             DOLLARTWO="'basename $2 .bst'";
44         else
45             DOLLARTWO="'dirname $2/'basename $2 .bst'";
46         fi
47         BST="${DOLLARTWO}";
48         shift 2;;
49     </script>
```

- `-e` or `--extra`: when we want to export all the entries of a `.bib` file, we can specify an extra `.bib` file that would contain entries that we don't want to export, but that are needed, *e.g.* for crossrefs.

```
50     (*script)
51         -e|--extra)
52         if [ "'dirname $2'" = "." ]; then
53             DOLLARTWO="'basename $2 .bib'";
54         else
55             DOLLARTWO="'dirname $2/'basename $2 .bib'";
56         fi
57         EXTRA="${EXTRA}${DOLLARTWO},";
58         EXTRABIB="${EXTRABIB},${DOLLARTWO}.bib";
59         shift 2;;
60     </script>
```

- `-o` or `--output`: the name of the output file.

```
61     (*script)
62         -o|--output-file)
63         if [ "'dirname $2'" = "." ]; then
64             DOLLARTWO="'basename $2 .bib'";
65         else
66             DOLLARTWO="'dirname $2/'basename $2 .bib'";
```

```

67         fi
68         OUTPUT="${DOLLARTWO}.bib";
69         shift 2 ;;
70     </script>

```

- `-c` or `--crossref` (or others): this options means that we want crossrefs to be included. Note that for any entry, field inheritance will be performed.

```

71     <*script>
72         -c|--crossref|--crossrefs|--with-crossref|--with-crossrefs)
73         CREF="1" ;
74         shift ;;
75     </script>

```

- `-n` or `--no-crossref`: don't include crossref'ed entries.

```

76     <*script>
77         -n|--no-crossref|--without-crossref)
78         CREF="20000" ;
79         shift ;;
80     </script>

```

- `-v` or `--version` for version number:

```

81     <*script>
82         -v|--version)
83         echo "2.10"; exit 0;;
84     </script>

```

- other dash-options are erroneous (except `-h`, but...):

```

85     <*script>
86         -*)
87         usage;;
88     </script>

```

- there should only remain file names: we add those names to the list of files.

```

89     <*script>
90         *)
91         if [ "`dirname $1`" = "." ]; then
92             DOLLARONE="`basename $1 ${EXT}`";
93         else
94             DOLLARONE="`dirname $1`/`basename $1 ${EXT}`";
95         fi
96         FILE="${FILE}${SPACE}${DOLLARONE}${EXT}";
97         if [ -z "${SPACE}" ]; then
98             SPACE=",";
99         fi;
100        shift;;
101    </script>

```

That's all folks:

```

102 <*script>
103     esac
104 done
105 </script>

```

### 2.1.3 The core of the script

We first set the name of the result and intermediary files:

```
106 ⟨*script⟩
107 FINALFILE=${OUTPUT};
108 if [ ! "${FINALFILE}" ]; then
109     FINALFILE="bibexport.bib";
110 fi
111 TMPFILE="bibexp.‘date +%s‘";
112 ⟨/script⟩
```

We then create the .aux file for the main run of Bib<sub>TEX</sub>. Note that this could call Bib<sub>TEX</sub>, with the expkeys.bst file, in the case where we want to export all entries of a .bib file but not crossrefs. Note how, in that case, we trick Bib<sub>TEX</sub> for inputting extra files twice: we include them with their short name first (with no extension), and then with the full name. We *need* to do that, since **string** abbreviations must be defined first, while crossrefs must occur after having been referenced.

```
113 ⟨*script⟩
114 if [ -z "${EXT}" ]; then
115     if [ -z "${EXTRA}" ]; then
116         cat > ${TMPFILE}.aux <<EOF
117 \citation{*}
118 \bibdata{${FILE}}
119 \bibstyle{${BST}}
120 EOF
121     else
122         cat > ${TMPFILE}.aux <<EOF
123 \citation{*}
124 \bibdata{${FILE}}
125 \bibstyle{expkeys}
126 EOF
127         bibtex -min-crossrefs=${CREF} ${TMPFILE} >/dev/null 2>&1;
128         mv -f ${TMPFILE}.bbl ${TMPFILE}.aux;
129         cat >> ${TMPFILE}.aux <<EOF
130 \bibdata{${EXTRA}${FILE}${EXTRABIB}}
131 \bibstyle{${BST}}
132 EOF
133     fi
134 else
135     cat ${FILE} | sed -e "s/bibstyle{.*/bibstyle{export}/" > ${TMPFILE}.aux;
136 fi
137 ⟨/script⟩
```

This was the hard part. We now call Bib<sub>TEX</sub>, clean and rename the output file, and remove intermediary files:

```
138 ⟨*script⟩
139 bibtex -min-crossrefs=${CREF} ${TMPFILE}
140 echo -e "Generated on ‘date +%c (%s).\n‘\n" > ${FINALFILE};
141 if [ ! "x${CREF}" = "x1" ]; then
142     sed -r -e \
143         "/^ *[cC][rR][oO][sS][rR][eE][fF] *= *[\^,]+,?$/d" \
144         ${TMPFILE}.bbl >> ${FINALFILE};
145 else
```

```

146   cat ${TMPFILE}.bbl >> ${FINALFILE};
147 fi
148 rm -f ${TMPFILE}.bbl ${TMPFILE}.aux ${TMPFILE}.blg
149 </script>

```

## 2.2 The expkeys.bst file

The only role of that file is to export the list of entries to be exported. It is used when we export all the entries of .bib files, except those of *extra* .bib files. Thus:

```

150 <*expkeys>
151 ENTRY{}{}{}
152 READ
153 FUNCTION{export.key}
154 {
155   "\citation{" cite$ "}" * * write$ newline$
156 }
157 ITERATE{export.key}
158 </expkeys>

```

## 2.3 The export.bst file

### 2.3.1 Some configuration values

```

left.width   We define here the indentation values, and the field delimiters. short width are
right.width  used for @preamble.
url.right.width 159 <*export>
left.short.width 160 FUNCTION{left.width}{#18}
right.short.width 161 FUNCTION{right.width}{#55}
left.delim 162 FUNCTION{url.right.width}{#61}
right.delim 163 FUNCTION{left.short.width}{#10} %% for @preamble
164 FUNCTION{right.long.width}{#63}
165 FUNCTION{left.delim}{"{"}
166 FUNCTION{right.delim}{"}"}
167 %FUNCTION{left.delim}{quote$}
168 %FUNCTION{right.delim}{quote$}
169 </export>

```

### 2.3.2 Entries

We use standard entries here. Of course, more entries could be added for special .bib files. Those extra entries will also have to be added in the main exporting function.

ENTRY

```

170 <*export>
171 ENTRY{
172 % Standard fields:
173   address
174   author
175   booktitle
176   chapter
177   edition
178   editor

```

```

179     howpublished
180     institution
181     journal
182     key
183     month
184     note
185     number
186     organization
187     pages
188     publisher
189     school
190     series
191     title
192     type
193     volume
194     year
195 % Special (but still somewhat standard) fields (natbib, germbib, ...):
196     abstract
197     doi
198     eid
199     isbn
200     issn
201     language
202     url
203 }{}{}
204 </export>

```

### 2.3.3 Basic functions

No comment.

```

or
and 205 <*export>
not 206 FUNCTION{not}
207 {
208     {#0}
209     {#1}
210     if$
211 }
212 FUNCTION{and}
213 {
214     'skip$
215     {pop$ #0}
216     if$
217 }
218 FUNCTION{or}
219 {
220     {pop$ #1}
221     'skip$
222     if$
223 }
224 </export>

```



### 2.3.4 Splitting strings

We design functions for splitting strings, so that the final .bib file will be cleanly indented.

```
space.complete
split.string 225 (*export)
  split.url 226 INTEGERS{left.length right.length}
  split name 227 STRINGS{ s t }
  228 FUNCTION{space.complete}
  229 {
  230   'left.length :=
  231   duplicate$ text.length$ left.length swap$ -
  232   {duplicate$ #0 >}
  233   {
  234     swap$ " " * swap$ #1 -
  235   }
  236   while$
  237   pop$
  238 }
  239 FUNCTION{split.string}
  240 {
  241   'right.length :=
  242   duplicate$ right.length #1 + #1 substring$ "" =
  243   {""}
  244   {
  245     's :=
  246     right.length
  247     {duplicate$ duplicate$ s swap$ #1 substring$ " " = not and}
  248     {#1 -}
  249     while$
  250     duplicate$ #2 <
  251     {
  252       pop$ " " s * ""
  253     }
  254     {
  255       duplicate$ s swap$ #1 swap$ substring$
  256       swap$
  257       s swap$ global.max$ substring$
  258     }
  259     if$
  260   }
  261   if$
  262 }
  263 FUNCTION{split.url}
  264 {
  265   'right.length :=
  266   duplicate$ right.length #1 + #1 substring$ "" =
  267   {""}
  268   {
  269     's :=
  270     right.length
  271     {duplicate$ duplicate$ s swap$ #1 substring$ "/" = not and}
  272     {#1 -}
```

```

273     while$
274     duplicate$ #2 <
275     {
276     pop$ " " s * ""
277     }
278     {
279     duplicate$ s swap$ #1 swap$ substring$
280     swap$ #1 +
281     s swap$ global.max$ substring$
282     }
283     if$
284     }
285     if$
286 }
287 FUNCTION{split.name}
288 {
289     'right.length :=
290     duplicate$ right.length #1 + #1 substring$ "" =
291     {""}
292     {
293     's :=
294     right.length
295     {duplicate$ duplicate$ s swap$ #5 substring$ " and " = not and}
296     {#1 -}
297     while$
298     duplicate$ #2 <
299     {
300     pop$ " " s * ""
301     }
302     {
303     #4 + duplicate$ s swap$ #1 swap$ substring$
304     swap$
305     s swap$ global.max$ substring$
306     }
307     if$
308     }
309     if$
310 }
311 </export>

```

### 2.3.5 Exporting fields

Here, we have four exporting functions, since we also have to deal with abbreviations:

```

field.export
abbrev.export 312 <export>
name.export 313 FUNCTION{field.export}
url.export 314 {
315     duplicate$ missing$
316     'skip$
317     {
318     left.delim swap$ * right.delim *
319     swap$

```

```

320     " " swap$ * " = " * left.width space.complete
321     swap$ "," *
322     {duplicate$ "" = not}
323     {
324         right.width split.string 't :=
325         *
326         write$ newline$
327         "" left.width space.complete t
328     }
329     while$
330 }
331 if$
332 pop$ pop$
333 }
334 FUNCTION{abbrev.export}
335 {
336     duplicate$ missing$
337     'skip$
338     {
339         swap$
340         " " swap$ * " = " * left.width space.complete
341         swap$ "," *
342         {duplicate$ "" = not}
343         {
344             right.width split.string 't :=
345             *
346             write$ newline$
347             "" left.width space.complete t
348         }
349         while$
350     }
351     if$
352     pop$ pop$
353 }
354 FUNCTION{name.export}
355 {
356     duplicate$ missing$
357     'skip$
358     {
359         left.delim swap$ * right.delim *
360         swap$
361         " " swap$ * " = " * left.width space.complete
362         swap$ "," *
363         {duplicate$ "" = not}
364         {
365             right.width split.name 't :=
366             *
367             write$ newline$
368             "" left.width space.complete t
369         }
370         while$
371     }
372     if$
373     pop$ pop$

```

```

374 }
375 FUNCTION{url.export}
376 {
377   duplicate$ missing$
378   'skip$
379   {
380     left.delim swap$ * right.delim *
381     swap$
382     " " swap$ * " = " * left.width space.complete
383     swap$ "," *
384     {duplicate$ "" = not}
385     {
386       url.right.width split.url 't :=
387       *
388       write$ newline$
389       "" left.width space.complete t
390     }
391     while$
392   }
393   if$
394   pop$ pop$
395 }
396 \</export>

```

### 2.3.6 Handling abbreviations

Abbreviations are difficult to deal with if we wish to still use them, since BibTEX will expand them before we can do anything. All we can do is to define them in a special way, in order to be able to get back to the abbreviations later on. This is precisely what we do:

```

jan-dec
acmcs-tcs 397 \<export>
remove.exports.from.months 398 MACRO{jan}{"export-jan"}
remove.export.from.journal 399 MACRO{feb}{"export-feb"}
400 MACRO{mar}{"export-mar"}
401 MACRO{apr}{"export-apr"}
402 MACRO{may}{"export-may"}
403 MACRO{jun}{"export-jun"}
404 MACRO{jul}{"export-jul"}
405 MACRO{aug}{"export-aug"}
406 MACRO{sep}{"export-sep"}
407 MACRO{oct}{"export-oct"}
408 MACRO{nov}{"export-nov"}
409 MACRO{dec}{"export-dec"}
410 MACRO{acmcs}{"export-acmcs"}
411 MACRO{acta}{"export-acta"}
412 MACRO{cacm}{"export-cacm"}
413 MACRO{ibmjrd}{"export-ibmjrd"}
414 MACRO{ibmsj}{"export-ibmsj"}
415 MACRO{ieeese}{"export-ieeese"}
416 MACRO{ieeetc}{"export-ieeetc"}
417 MACRO{ieeetcad}{"export-ieeetcad"}
418 MACRO{ipl}{"export-ipl"}

```

```

419 MACRO{jacm}{"export-jacm"}
420 MACRO{jcss}{"export-jcss"}
421 MACRO{scp}{"export-scp"}
422 MACRO{sicomp}{"export-sicomp"}
423 MACRO{tocs}{"export-tocs"}
424 MACRO{tods}{"export-tods"}
425 MACRO{tog}{"export-tog"}
426 MACRO{toms}{"export-toms"}
427 MACRO{toois}{"export-poois"}
428 MACRO{toplas}{"export-toplas"}
429 MACRO{tcs}{"export-tcs"}
430 INTEGERS{ intxt }
431 FUNCTION{remove.exports.from.months}
432 {
433   #0 'intxt :=
434   duplicate$ missing$
435   'skip$
436   {'t :=
437   ""
438   {t #1 #1 substring$ "" = not}
439   {
440     t #1 #7 substring$ "export-" =
441     {intxt
442       {right.delim * #0 'intxt :=}
443       'skip$
444       if$
445       duplicate$ "" =
446       'skip$
447       {" # " *}
448       if$
449       t #8 #3 substring$ *
450       t #11 global.max$ substring$ 't :=}
451     {intxt
452       'skip$
453       {duplicate$ "" =
454         {}
455         {" # " *}
456         if$
457         left.delim * #1 'intxt :=}
458         if$
459         t #1 #1 substring$ *
460         t #2 global.max$ substring$ 't :=}
461       if$
462       }
463     while$
464     intxt
465     {right.delim *}
466     'skip$
467     if$
468     }
469   if$
470 }
471 FUNCTION{remove.export.from.journals}
472 {

```

```

473 duplicate$ missing$
474   'skip$
475   {
476     duplicate$ #1 #7 substring$ "export-" =
477       {#8 global.max$ substring$}
478     {left.delim swap$
479       right.delim * *}
480     if$
481   }
482 if$
483 }
484 </export>

```

### 2.3.7 Now, we export...

We gather everything. This is where special fields must be added for being exported:

```

entry.export.standard
  entry.export.extra 485 <*export>
    entry.export      486 FUNCTION{entry.export.standard}
      export          487 {
488       "address" address field.export
489       "author"  author name.export
490       "booktitle" booktitle field.export
491       "chapter" chapter field.export
492       "crossref" crossref field.export
493       "edition" edition field.export
494       "editor"  editor name.export
495       "howpublished" howpublished field.export
496       "institution" institution field.export
497       "journal" journal remove.export.from.journals abbrev.export
498       "key" key field.export
499       "month" month remove.exports.from.months abbrev.export
500       "note" note field.export
501       "number" number field.export
502       "organization" organization field.export
503       "pages" pages field.export
504       "publisher" publisher field.export
505       "school" school field.export
506       "series" series field.export
507       "type" type field.export
508       "title" title field.export
509       "volume" volume field.export
510       "year" year field.export
511     }
512 FUNCTION{entry.export.extra}
513 {
514   "abstract" abstract field.export
515   "doi" doi field.export
516   "eid" eid field.export
517   "isbn" isbn field.export
518   "issn" issn field.export
519   "language" language field.export
520   "url" url url.export

```

```

521 }
522 FUNCTION{entry.export}
523 {
524   entry.export.standard
525   entry.export.extra
526 }
527 FUNCTION{export}
528 {
529   "@" type$ * "{" * cite$ * "," * write$ newline$
530   entry.export
531   "}" write$ newline$ newline$
532 }
533 </export>

```

### 2.3.8 Miscellanea

We also have to handle preamble, and to define functions for each entry type (we won't use them but otherwise, Bib<sub>T</sub><sub>E</sub><sub>X</sub> would complain).

```

preamble
  header 534 <*export>
entries.headers 535 FUNCTION{preamble}
article-unpublished 536 {
537 preamble$ duplicate$ "" =
538 'pop$
539 {
540   ",-----." write$ newline$
541   "|   PREAMBLE   |" write$ newline$
542   "'-----'" write$ newline$ newline$
543   "@preamble{ " swap$
544   quote$ swap$ * quote$ *
545   {duplicate$ "" = not}
546   {
547     right.long.width split.string 't :=
548     *
549     write$ newline$
550     "" left.short.width space.complete t
551   }
552   while$
553   "}" write$ newline$ newline$
554   pop$ pop$
555 }
556 if$
557 }
558 FUNCTION{header}
559 {
560 %"*** This file has been automatically generated by bibexport ***"
561 %write$ newline$
562 %"*** See http://www.lsv.ens-cachan.fr/~markey/bibla.php ***"
563 %write$ newline$
564 %"*** for more informations about bibexport. ***"
565 %write$ newline$
566 newline$
567 }

```

```

568 FUNCTION{entries.header}
569 {
570     ",-----." write$ newline$
571     "| BIBTEX ENTRIES |" write$ newline$
572     "'-----'" write$ newline$ newline$
573 }
574 FUNCTION{article}{export}
575 FUNCTION{book}{export}
576 FUNCTION{booklet}{export}
577 FUNCTION{conference}{export}
578 FUNCTION{habthesis}{export}
579 FUNCTION{inbook}{export}
580 FUNCTION{incollection}{export}
581 FUNCTION{inproceedings}{export}
582 FUNCTION{journals}{export}
583 FUNCTION{manual}{export}
584 FUNCTION{mastersthesis}{export}
585 FUNCTION{misc}{export}
586 FUNCTION{phdthesis}{export}
587 FUNCTION{proceedings}{export}
588 FUNCTION{techreport}{export}
589 FUNCTION{unpublished}{export}
590 </export>

```

### 2.3.9 Main program

We now can execute and iterate those functions:

```

591 <*export>
592 READ
593 EXECUTE{header}
594 EXECUTE{preamble}
595 EXECUTE{entries.header}
596 ITERATE{export}
597 </export>

```