

Contents

Preface	2
Introduction	2
Interface Fluent-Scheme	2
RP-variables	3
CX-variables	3
Interface Fluent-Scheme-UDFs.....	3
Data exchange	3
Calling Functions.....	4
Arithmetic Functions	4
Global variables Scheme.....	4
Local Scheme Variables	5
Lists	5
if command.....	5
do-loop.....	6
format command.....	8
for-each loop.....	8
Aliases in TUI.....	9
Example: Creating animation	9
Example: report data from data files.....	10
Example: Getting values from data files or case	13
Example: Exporting fluent zone names for UDF.....	14
Iteration Control	16
Peculiarities of the Fluent Scheme	17
List command.....	17
Format command	17
System command	17
Fluent variables and functions.....	17
Literature on scheme.....	17
Fluent Scheme standard functions	18
Fluent-Scheme Environment	19

Preface

Scheme offers many ways to automate processes in Fluent to expire. Unfortunately, there is up to today virtually no documentation on this topic by Fluent. Dybvik's Scheme book (see bibliography) has really helped me to learn to understand Scheme. However for use in Fluent is needed some knowledge more than the standard Scheme. To my experience with Fluent Scheme also provided to others, I started in September 2000, to write this script. The script is somewhat rudimentary, but still better than nothing.

In the meantime, a few enhancements to come, I would like to thank at this point Zoran Carija for the tip with the with-output-to-file function and Angelo Sozzi on the effort to translate these notes to English.

I am pleased again and again with received positive feedback about this script and that even FLUENT Germany recommends this notes to its customers. FLUENT said, on information from FLUENT Germany, that no official Scheme documentation will be still released, because Scheme in the future will be replaced by Python as official scripting language.

Mirko Javurek, Linz am 27. 10. 2004

Introduction

Scheme is a dialect of LISP;

Uniform Code Format:

```
(command_name argument1 argument2 ...)
```

Each command call is a function call and therefore deliver a result. Command and variable names are not case-sensitive (should only contain small letters) must begin with a letter and can, beside a-z and 0-9, the special +*/<>=?.:%\$!~^_ include.

Comments start with;; and end at the end of line.

Interface Fluent-Scheme

Calling Scheme commands in Fluent:

Input code in the command-line interface (also using the mouse to copy the commands from other Fluent windows – for example editor - on X-Selection in Unix is possible), or

Write Scheme program with a text editor, save (. Scm extension) and read in Fluent-menu "File / Read / Scheme";

If there is a Scheme file named .fluent in the home folder it will start with Fluent automatically.

In the menu "Solve / monitor / Commands / Command" is possible to set text and Scheme commands and these will then at each iteration or each time-step be executed;

Calling Fluent commands in Scheme:

Text Interface command:

```
(ti-menu-load-string "display / temperature contour 30 100)
```

Return value: # t if successful, # f if errors or cancel by pressing Ctrl-C, Ctrl-C stops Fluent command, but not Scheme programs!

GUI Command: Journal of the desired recorded GUI actions, Journal that contains direct Scheme commands such as:

```
(cx-gui-do cx-activate-item "Velocity Vectors * * PushButton1 panel buttons (OK)")
```

Text Interface commands are faster, more compact and have a versatile use. GUI commands are slow, confusing and poorly adapted (reference list entry, for example are not by name of the item, but number). Text Interface commands are therefore preferable than GUI commands; GUI commands are used only when there is no text command available.

Text Interface commands are not commented (Ab Fluent 5.5?). Procedure: Search the desired text command, try it and try all the entries together.

Output of Scheme at Fluent text Interface:

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(display object)

(newline)

File access (read / write) in Scheme (see examples).

RP-variables

Show variable, such as current simulation time:

```
> (Rpgetvar 'flow-time)
```

0.1

Setting variable:

```
> (Rpsetvar 'flow time 0)
```

All RP variables are defined in the case file (Section "Variables").

CX-variables

Show variable, eg: color progression table:

```
> (cxgetvar 'CMAP-list)
```

0.1

Variable:

```
> (Cxsetvar 'def CMAP "rgb")
```

All CX-variables are (partially?) In the case file is defined (Section "Cortex Variables").

Interface Fluent-Scheme-UDFs

Data exchange

It is possible to create your own RP-variables, these can be addressed in Fluent text interface and in specific functions in UDF.

Definition of an new RP-variable:

```
(rp-var-define name default-init-and-type # f value)
```

types: 'int' real 'boolean' string ...?

For example:

```
> (Rp-var-define 'udf/var1 0' real # f)
```

Information about Variable:

```
> (Rp-var objectClass' udf/var1)
```

```
(real udf/var1 0 # f 0)
```

```
> (Rp-var objectClass' udf/var2)
```

```
# f
```

It is possible to change and query a variable as above with rpsetvar and rpgetvar.

If a RP-variable is once defined, they will remain until quitting Fluent receive (!), In each case file saved, and when such a Case Files is loaded - if not defined – it will generate the variable with the stored value .

In UDFs it is possible to define RP-variables with the C functions (declared in Fluent.Inc / fluentX.Y / src / var.h)

```
real RP_Get_Real (char * s);
```

```
long RP_Get_Integer (char * s);
```

```
RP_Get_String char * (char * s);
```

```
boolean RP_Get_Boolean (char * s);
```

```
RP_Set_Real void (char * s, real v);
```

```
RP_Set_Integer void (char * s, long v);
```

```
RP_Set_Boolean void (char * s, boolean v);
```

```
RP_Set_String void (char * s, char * v);
```

```
RP_Set_Symbol void (char * s, char * v);
```

query and setting respectively , for example:

```
var1 = RP_Get_Real ( "udf/var1");  
RP_Set_Real ( "udf/var1", 3.2);
```

For UDFs in parallel mode specific rules to access RP-variables apply, see FLUENT UDF manual.

Calling Functions

UDFs of type EOD can be called by Scheme via the command

```
(% udf-on-demand "udf-eod-name)
```

There is currently no way known to call Scheme functions from a UDF, the C function

```
CX_Interpret_String ( "scheme-command-string") - declared in Fluent.Inc / fluentX.Y / cortex / src / cx.h –  
Is interpreted in the "scheme-command-string", but has no access to the Environment.
```

Arithmetic Functions

Basic functions + - * /, in RPN mode, more than 2 arguments are possible:

```
> (+ 2 4 5)
```

```
11
```

```
> (/ 6 3)
```

```
2
```

```
> (/ 2) ;; is (/ 1 2)
```

```
0.5
```

Furthermore (abs x), (sqrt x), (expt xy) [= xy], (exp x) [= ex], (log x) [= ln x], (sin x), (cos x), (atan x), (atan y x) [= arctan (x / y)], ...

Integer (!)-Functions:

```
> (Remainder 45 6)
```

```
3
```

```
> (Modulo 5 2)
```

```
1
```

(truncate x) (round x), (ceiling x) (floor x), ...

furthermore

(max x y ...), (min x y ...)

e.g. from a list search for maximum:

```
> (Apply max (1 5 8 3 4))
```

```
8
```

and some others (see Scheme literature).

Global variables Scheme

Define with:

```
> (Define x 3)
```

```
> (+ x 1)
```

```
4
```

No distinction between variable types (Integer, Real, String, ...) - Each variable can accept values of each type.

Value change with renewed definition (not possible within functions, there exists a field for local variables, so that the variable with local definition will have a new definition) or better

```
(set! x 1)
```

Value displayed with

```
(display x)
```

or

(write x)

Write should be used only if Fluent variables are in a saved and later loaded file. Write displays Strings with quotes.

Constants: integer (2), float (2.5), boolean (# t for true, false # f) Strings ("this is a text string") and symbols:

'symbol, eg:

(define x 'this-is-a-symbol)

Special characters for string definitions:

\ ""

\n new line

Global variables and self-defined scheme functions will remain until quitting Fluent.

Local Scheme Variables

(let ((var1 value1) (var2 value2) ...)

... commands in the validity range ...

)

Lists

Definition example:

> (Define my-surfaces' (wall-wall top-bottom symmetry))

Arbitrary length, dynamic allocation, nesting is possible.

Define lists with '(elements ...):

> (Define l '(a b c))

First item of a list

> (Car 1)

a

"Rest" of a list (list without first element)

> (Cdr 1)

(b c)

Number of list elements

> (Length l)

3

i-th element of a list

(listref list i)

Search for element in list:

> (Member 'd' (a b c d e f g))

(d e f g)

Using function in list(s):

> (Map (lambda (x) (x * x)) '(1 2 3))

(1 4 9)

> (apply + '(2 4 5))

11

> (apply max (1 5 8 3 4))

8

if command

if-command is a function:

(if cond true-value false-value)

cond is a boolean expression that is either # t (true) or # f (false) is.

Operations comparison:

Equality:

(= a b) ;; number

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(eq? a b) ;; objects
(eqv? a b) ;; same value objects

Relations:

(positive? x)

(negative? x)

(< a b)

(> a b)

(<= a b)

(> = a b)

Boolean functions:

(not a)

(and a b c ...)

(or a b c ...)

Extension of the "if" - and "else" branch for more commands with block command "begin" ("sequencing", generally applicable):

(if cond

(begin ;; if

...

True-value

)

(begin ;; else

...

False-value

)

)

If the result value of the if-command is not needed, you can the "else" branch and the result values are omitted.

More complex commands for conditional flow control (eg, for piecewise defined functions):

(cond (test1 value1) (test2 value2) ... (else value))

and for discrete values of a variable

(case x ((x11 x12 x13 ...) value1) ((x21 x22 x23 ...) value2) ... (else value))

If x in one of the lists found (eg in (x11 x12 x13 ...)), then the corresponding value is returned (value1).

do-loop

Simplest form (variable, start value, the value of the loop variable after each loop pass that will be assigned, termination condition):

(do ((x start-x (+ x delta-x))) ((> x x-end)) ... loop body ...)

Multiple or no loop variables are possible.

Example liso-surfaces creation: more iso-surfaces should be generated in uniform intervals of iso-values and automatically named. First, the dialogue in TUI for generating an iso-surface will be collected:

>

```
Adapt/      grid/      surface/
display/plot/      view/
define/      report/    exit/
file/        solve/
```

> Surface

/ surface>

```
delete-surface      mouse-line      point-array
surface-cells       mouse plane     rake-surface
iso-surface         mouse rake      rename-surface
```

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

iso-clip	partition-surface	sphere-slice
list-surfaces	plane slice	zone-surface

/ surface> iso-surface

```
iso-surface of>
pressure          entropy          x-surface-area
pressure coefficient  total-energy      y-surface-area
dynamic-pressure   internal-energy   z-surface-area
...
rel-total-temperature  x-coordinate     dp-dx
wall temp-out-surf    y-coordinate     dp-dy
wall-temp-in-surf     z-coordinate     dz-dp
```

```
iso-surface of> x-coordinate
new surface id / name [x-coordinate-31] test name
range [-10.0131, 4.8575001]
surface from [()] ()
()
iso-value (1) (m) [()] 1.234
iso-value (2) (m) [()] ()
One TUI command is also possible (all entries in a row together):
surface / iso-surface x-coordinate test name () 1,234 ()
```

```
This "parametric" Scheme-loop:
(do ((x 0 (+ x 0.2))) (> x 3.1))
(ti-menu-load-string
  (format # f "surface / iso-surface x-coordinate x-3.1f ~ () ~ a () 'xx))
)
```

```
Creates the following text interface commands:
surface / iso-surface x-coordinate x-0.0 () 0 ()
surface / iso-surface x-coordinate x-0.2 () 0.2 ()
surface / iso-surface x-coordinate x-0.4 () 0.4 ()
...
surface / iso-surface x-coordinate x-3.0 () 3 ()
```

```
Refinement: better names for positive and negative coordinates:
(do ((z -1 (+ z 0.25))) (> z 1))
(ti-menu-load-string (format # f "surface / iso-surface z-coordinate z ~ a ~ 05.3f () ~ a () "
(if (> z = 0)" + "" " ) zz))
)
```

```
surface / iso-surface z-coordinate z-1,000 () -1 ()
surface / iso-surface z-coordinate z-0,750 () -0.75 ()
surface / iso-surface z-coordinate z-0,500 () -0.5 ()
surface / iso-surface z-coordinate z-0,250 () -0.25 ()
surface / iso-surface z-coordinate z +0,000 () 0 ()
surface / iso-surface z-coordinate z +0,250 () 0.25()
surface / iso-surface z-coordinate z +0,500 () 0.5 ()
surface / iso-surface z-coordinate z +0,750 () 0.75 ()
surface / iso-surface z-coordinate z +1,000 () 1 ()
```

Amendment: 2 loop variables:

```
(do ((x 0 (+ x 0.2)) (i 1 (+ i 1))) (> x 3.1))
(ti-menu-load-string
(format # f "surface / iso-surface x-coordinate ~ x-02d () ~ a () 'ix))
)
```

```
surface / iso-surface x-coordinate x-01 () 0 ()
surface / iso-surface x-coordinate x-02 () 0.2 ()
surface / iso-surface x-coordinate x-03 () 0.4 ()
...
surface / iso-surface x-coordinate x-16 () 3 ()
```

format command

(format # f "format string as in C printf with patterns for var1, var2, ..." var1 var2 ...)

Instead, of the %-character in C the tilde (~) controls a pattern ;

Pattern examples:

```
~ a    random variable in general format (string without "")
~ d    integer number
~04d   integer with zeros on the front is always 4 digits fill ( 5 is about 0005), eg this is important for file names.
~f     floating-point number
~4.2f  floating point number, a total of 4 characters long, 2 digits after the decimal: 1.2 will be 1.20
~s     string in "" include: from (format # f "string: ~ s !" "text"), the string: "text"! ... and others?
```

Special characters:

```
\n line feed
\"  "
```

The format command and its patterns are not included in the Scheme standard, are therefore used in Fluent Scheme implementation dependently, it is unfortunately not documented

for-each loop

Performs a self-defined function for each element of one or more lists:

```
(for-each function list1 list2 ...)
```

The number of function arguments of "function" must correspond to the number of lists.

Suitable e.g. for: Fluent zone names or IDs, filenames (if they do not contain capital letters),

Example, SETTING temperature and wall velocity for several BC wall zones:

```
(define velocity 0.1)
(for-each
(lambda (zone)
(ti-menu-load-string
(format # f "def / bc / wall ~ a 0 0 yes yes giesspulver temperature 1800 yes no no no ~ a 0 -1 0 NO 0 0.5 "velocity
zone)
)
(newline) (display "      ")
)
'(
kok_li kok_re
kok_innen kok_aussen
bieg_li biege_re
bieg_aussen biege_innen
```

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br


```

kreis_li kreis_re
kreis_aussen kreis_innen
)
)

```

Lambda Code to define "local" functions:
(lambda (arg1 arg2 ...) ... function value)

Aliases in TUI

In the TUI abbreviations can be created:

```
(alias 'name scheme-function)
```

For example:

```
(alias 'time (lambda () (display (rpgetvar' flow-time))))
```

Call interface in the text:

```
> Time
```

```
0.1
```

Arguments cannot be passed directly to Scheme functions (always zero arguments, ie lambda ()), but they must be inputted by the following functions of the text interface:

```
(read-real prompt default)
```

```
(read-integer prompt default)
```

```
(ti-read-unquoted-string prompt default)
```

```
(yes-or-no? prompt default)
```

prompt is a string, default and the default value that is returned when the user just presses return.

Aliases are always automatically available when defined in the . Fluent-file written (see above).

Example: Creating animation

From the data files of a transient simulation single frames will be created for an animation. The names of the data files are numbered, with initial, final value and certain step size. Errors encountered during the execution of a Fluent command, or a termination by Ctrl-C will also cause the Scheme program to quit.

```
(define datfilename "test")      ;; -> 0010.dat test, test-020.dat ...
```

```
(define first-index 10)
```

```
(define last-index-110)
```

```
(define delta 10)
```

```
(define imagefilename "image");; -> image-01.bmp ...
```

```
(define (time) (rpgetvar 'flow-time))
```

```
(define t0 0)
```

```
;;-----
```

```
;; function, the frames created for the film
```

```
;;-----
```

```
(define (pp)
```

```
(let
```

```
(
```

```
(break # f)
```

```
)
```

```
(ti-menu-load-string "display / set / hardcopy / driver / tiff");; TIFF format
```

```
(ti-menu-load-string "display / set / hardcopy / color-mode / color ");; default is " gray "
```

```
(do      ((j first-index (j + delta));; datfile start value and delta
```

```
(i 1 (+ i 1)));; imagefile start value and delta
```

```
((or (> j last-index) break));; datfile final value
```

```
(set! break (not (and
```

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

```
(ti-menu-load-string
(format # f "file / read-data ~ a ~ 04d.dat" datfilename j))
(begin (if (= i 1) (set! t0 (time))) # t)
(dispatch (system "rm temp . tif ");; hardcopy does not work if file already exists
(ti-menu-load-string "display / hardcopy temp.tif ")
(system
(format # f "convert temp.tif ~ a ~ 02d.bmp &" imagefilename i))
;; convert-command of www.imagemagick.com
)))
)
(if break (begin (newline) (newline) (display "scheme interrupted!") (newline)))
)
)
```

Simple example (disp)-function: contour-plot:

```
(define (disp)
(ti-menu-load-string "display / contour / temperature 290 1673")
)
```

Example (disp)-Function: overlay Contours / velocity-vectors, own Show Time:

```
(define (disp)
(and
(ti-menu-load-string
(format # f "displaylang set title \" Time 5.1fs = ~ \ " (- (time) t0))
(ti-menu-load-string "display / set / overlays no")
(ti-menu-load-string "display / temperature contour 290 1673")
(ti-menu-load-string "display / set / yes overlays")
(ti-menu-load-string "displaylang / velocity vectors, velocity-magnitude 0.0 1.0 5 0")
;; colored by min max scale skip
)
)
```

Example (disp)-function: Iso-surface generated from this phase boundary VOF with y-coordinate (= height) tinted:

```
(define (disp)
(and
(ti-menu-load-string "display / surface / iso-surface interface vof-steel-1, 0.5, ')
(ti-menu-load-string" display / set / CONTOUR / surface interface-1 () ")
(ti-menu-load-string" display / contour y-coordinate 2,755 2,780)
(ti-menu-load-string "display / surface / delete interface-1")
)
)
```

Calling the (disp)-function for testing:

```
> (disp)
```

Calling the function to generate the images:

```
> (pp)
```

Example: report data from data files

Reported data must be in a transcript file to be written:

```
(ti-menu-load-string "file / start-transcript temp.trn")
(ti-menu-load-string "report / cell-average fluid, temperature ')
(ti-menu-load-string" file / stop -transcript ")
```

There is alternatively a separate Scheme function:

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

```
(with-output-to-file "temp.trn "  
(lambda ()  
(ti-menu-load-string "report / cell-average fluid, temperature ")))
```

With this there is no output on the screen.

Transcript file "temp.trn":

```
report / cell-average fluid, temperature  
volume-average of temperature on cell zones (fluid)  
volume-weighted average = 300  
file / stop-transcript
```

read transcript-file in Scheme as a list of objects ("data"), after "=" Search, and later for the following element of the required number of value:

```
(let  
(  
(data  
(let ((p (open-input-file "temp.trn")))  
(let f ((x (read p)))  
(if (eof-object? x)  
(begin  
(close-input-port p)  
'())  
(cons x (f (read p)))  
)  
)  
)  
(Value 0)  
)  
(Ti-menu-load-string "! Temp.trn rm") (newline)
```

```
(do ((i 0 (+ i 1)) (>= i (length data)))  
(if (eq? (list-ref data i) '=)  
(set! value (list-ref data (+ i 1)))  
)  
)  
Value  
)
```

More elegant and shorter version of the do-loop with no variable value is needed:

```
(cadr (member '= data))
```

Nested car-cdr commands:

```
(cadre x) = (car (cdr x))
```

For heat flux balance (TUI is created for all surfaces) heat flux for specific surfaces are plotted

Format of the balance plot:

...

Area 15 (steel plate): 11.2

Area 5 (steel mold): 53.5

Area 6 (slag-exterior): 32.4

Area 14 (slag-hood): 26.9

...

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

Scheme program:

```
(let
  (
    (p (open-output-file "fluxes.txt"))      ;; open output text file
    (n 0)
    (surfaces '(
      stahl-bodenplatte
      stahl-kokille
      stahl-schlacke
      haube-schlacke
      elektrode-schlacke
      schlacke-innen
      schlacke-aussen
      aufsatz-schlacke
      haube-kuehlung
    ))
  )
  (For-each
    (lambda (filename)
      (if (zero? (Modulo n 2)) ;; only take every second datafile
        (begin
          (ti-menu-load-string
            (format # f "file read-data ~ a" filename))
          (ti-menu-load-string "file / start-transcript temp.trn")
          (ti-menu-load-string "report / heat-transfer")
          (ti-menu-load-string "file / stop-transcript" )
          (define data      ;; transcriptfile in "data" download
            (let ((p (open-input-file "temp.trn")))
              (let f ((x (read p)))
                (if (eof-object? x)
                  (begin
                    (close-input-port p)
                    '())
                  (cons x (f (read p)))
                )
              )
            )
          )
          (ti-menu-load-string "! temp.trn rm")
        )
      )
    )
    (display (time) p)
    (display " " p)
    (for-each
      (lambda (zone)
        (begin
          (display (list-ref (member (list zone) data) 2) p) ;; fluxwert area identified by (display "" p)
        )
      )
      surfaces
    )
  )
)
```

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

```

(newline p)
)
)
(Set! N (n + 1)
)
'(
best-0060.dat best-0120.dat best-0132.dat best-0144.dat best-0156.dat
best-0168.dat best-0180.dat best-0192.dat best-0204.dat best-0216.dat
best-0228.dat best-0240.dat best-0252.dat best-0264.dat best-0276.dat
best-0288.dat best-0300.dat best-0312.dat best-0324.dat best-0336.dat
)
)
(close-output-port p)
)

```

Data-file-list can be generated at UNIXs shell with `x-ls *. dat`.

Example: Getting values from data files or case

Fluent format schema files are nested lists:

Datafile example:

```

(0 "fluent5.3.18")

(0 "Machine Config:")
(4 (23 1 0 1 2 4 4 4 8 4 4))

(0 "grid size")
(33 (10540 21489 10947))
(0 "Variables:")
(37 (
(flow-time 3.7)
(time-step 0.1)
(periodic / pressure-derivative 0)
(number-of-samples 0)
(dpm / summary ())))

(0 "Data:")
(2300 (1 1 1 0 0 1 430))
...

```

They can therefore be read very simply as Scheme objects. For example, read time from data files and rewrite them with time code hh: mm: ss

```

(let ((p (open-input-file filename)) (found # f) (t -1))
  (do ((x (read p) (read p))) ((or found (eof-object? x)) (close-input-port p))
    (if (eqv? (car x) 37) ;; variables
        (begin
          (for-each
           (lambda (y)
             (if (eqv? (car y) 'flow-time)
                 (begin
                  (set! found # t)

```

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

```

(set! t (cadr y))
)
)
)
(cadr x)
(newline)
)
)
)
(Ti-menu-load-string (format # f "! Mv ~ a ~ a ~ a.dat" filename newdatname (sec-> hms t)))
)
Function sec-> hms converts seconds in hh: mm: ss format:
(define (sec-> hms t)
(let *
(
(H (truncate (/ t 3600))) (t1 (- t (h * 3600)))
(m (truncate (/ t1 60)))
(s (truncate (- t1 (m * 60))))
)
(Format # f "~ 02d ~ 02d ~ 02d" h m s)
)
)
)

```

Example: Exporting fluent zone names for UDF

In UDFs s BC-zone can be define as ID thread_id (t) and as type THREAD_TYPE (t), but the name will not be used. The simpler variant creates a data estructure with the following Scheme function, that can be copied in the UDF code:

```

(define (export-bc-names)
(for-each
(lambda (name)
(display
(format # f " {~a, \"~a\", \"~a\"},\n"
(zone-name->id name)
name
(zone-type (get-zone name)))
)))
(inquire-zone-names)
)
)

```

Fluent output:

```

(export-bc-names)
{26, "wall-wehr-l-shadow", "wall"},
{2, "fluid", "fluid"},
{29, "wall-damm-l-shadow", "wall"},
{15, "wall-damm-l", "wall"},
{17, "inlet", "mass-flow-inlet"},
{25, "default-interior", "interior"}
...

```

this text must be copied in the followinf UDF code:

```
#define nc 100
```

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

```

typedef struct zone_info_struct
{
int id;
char name[nc];
char type[nc];
}
zone_info;
zone_info zone[]={
/** here is the text copied from fluent */
{26, "wall-wehr-l-shadow", "wall"},
{2, "fluid", "fluid"},
{29, "wall-damm-l-shadow", "wall"},
{15, "wall-damm-l", "wall"},
{17, "inlet", "mass-flow-inlet"},
{25, "default-interior", "interior"}
...
};
#define n_zones (sizeof(zone)/sizeof(zone_info))

```

Now you can enter the zone name in the UDF code zone [i]. The name to be addressed.

The alternative is a Scheme function that writes the zones as a string in a RP-variable. The advantage is that the RP-variable will be saved in the case file, and then you can create several case files with their own UDF functions without the need to re-compile this one. The Scheme function must be called only once when setting the case file. Disadvantage: RP-variables are not so easy to work in the parallel solvers.

```

(define (bc-names->rpvar)
(let ((zone-data ""))
(for-each
(lambda (name)
(set! zone-data
(format #f "~a ~a ~a ~a " zone-data
(zone-name->id name)
name
(zone-type (get-zone name))
)))
(inquire-zone-names)
)
(display zone-data)
(rpsetvar* 'zone-names 'string zone-data)
)
)

```

The previous function will use:

```

(define (rpsetvar* var type value) ;; create cortex variable if undefined
(if (not (rp-var-object var))
(rp-var-define var value type #f)
(rpsetvar var value)
)
)

```

UDF-Code:

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

```

#define max_n_zones 200
#define max_zonenamechars 200
/* global variable */
char zone_name[max_n_zones][max_zonenamechars];
char zone_type[max_n_zones][max_zonenamechars];
#define THREAD_NAME(t) zone_name[THREAD_ID(t)]
/* local variable */
char *rps,s[1000];
int i,n;
for(i=0; i<max_n_zones; i++) zone_name[i][0]=0; /* initialisieren */
rps = RP_Get_String("zone-names");
while(rps[0])
{
  sscanf(rps,"%s%n", s,&n); rps += n;
  i = atoi(s);
  sscanf(rps, "%s%s %n", zone_name[i], zone_type[i],&n); rps += n;
}

```

There is even a macro `THREAD_NAME (t)`, with the name of the zone that can be addressed.

Iteration Control

Especially interesting for unsteady calculations;

Cortex variables (can also be set!) :

Time (t):

flow-time

Number of the current time step (N):

time-step

Size of time step (Δt):

physical-time step

List of saved iterations (current iteration first, then the previous, etc.)

(residual-history "iteration")

that the current Iterationszahl:

(car (residual-history "iteration"))

Lists of the residuals (entries according to the "iteration" entries):

(residual-history "continuity")

(residual-history 'x-velocity ")

(residual-history "temperature")

...

TUI commands to iterate (steady or unsteady, when unsteady more input is necessary):

solve / iterate number of iterations

Transient:

solve / dual-time-iterate number of time steps max iterations

Step size must e.g. with

(rpsetvar 'physical-time-step 0.1)

be set.

Using these commands and variables complex iterations can be programmed, for example

- unsteady: identification of computing time interval to calculate the number of time steps;
- automatic continuation of the unsteady after stop
- variable step size according to table (eg 10 seconds $dt = 0.1$ s, then 20 seconds $dt = 0.2$ s, etc.);

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

- Auto-Save at certain times of the calculation with time code in the file name (data-00: 10.dat) constant time gaps despite of variable increment
- own adaptive step size control
- maintain a backup data file for lengthy and for crash risky calculations: all x iteration
Save new data file, and then auto-delete the previously stored data file.

Peculiarities of the Fluent Scheme

eval command and environment

(eval expression environment)

The Standard Scheme Environment (the-environment) includes all the symbols, in addition to the defined Scheme standard symbols, are also variables and functions from the user (define ...) and from Fluent (!). If the interpretation of an expression is not required, you can empty the list '() or use #f. Example:

(define x 3)

(define y '(+ x 2)) ; y is a list with elements +, x, 2

(eval y (the-environment)) ; list y will be interpreted as a scheme command; Answer: 5

The following functions can be used to test whether a symbol is defined (bound) and whether a value is assigned (assigned):

(symbol-bound? 'symbol (the-environment))

(symbol-assigned 'symbol (the-environment))

List command

(list-head list n)

(list-tail list n)

Format command

Have a look at section „format command“ pag 7.

System command

Run a shell command, such as:

(system "rm temp.jou")

Fluent variables and functions

All Fluent variables and functions are in the environment (the-environment) defined, even if not documented. See section "Fluent Scheme Environment".

Literature on scheme

Unfortunately, there is no literature relating specifically to Fluent Scheme. Scheme is a very powerful language, with it demanding applications such as artificial intelligence can be conceived, most Scheme books go much more depth than what is necessary for Fluent. I use this book:

- R. Kent Dybvig, The Scheme Programming Language, Second Edition, 1996 Prentice Hall PTR.

Fluent recommends the following scheme links on the Web:

- [Http://www.swiss.ai.mit.edu/projects/scheme](http://www.swiss.ai.mit.edu/projects/scheme)

- <http://www.schemers.org/>

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

Some examples of FLUENT are now available at the FLUENT User Services Center in the "Online Technical Support":
- [Http://www.fluentusers.com/](http://www.fluentusers.com/)

Fluent Scheme standard functions

The following functions are part of the standard Fluent Scheme and do not appear on the Environment. I have extracted this list from the Fluent program file, but I have to conclude that it is, unfortunately, incomplete: there are functions that are not in this list that are included in the Environment (eg, list-head, tail and list-list-remove). Meanwhile, I have a few of these missing functions added (10-2004).

<	close-input-port	format
<=	close-output-port	format-time
=	closure?	gc
>	closure-body	gc-status
>=	cond	general-car-cdr
-	cons	getenv
/	continuation?	hash-stats
*	copy-list	if
+	cos	int
abs	cpu-time	integer?
access	debug-off	integer->char
acos	debug-on	interrupted?
and	define	lambda
append	display	length
append!	do	let
apply	dump	let*
asin	echo-ports	list
assq	env-lookup	list-head
assv	eof-object?	list->string
atan	eq?	list-tail
atan2	equal?	list->vector
begin	eqv?	local-time
bit-set?	error	log
boolean?	error-object?	log10
call/ccinput-port?	err-protect	logical-and
car	err-protect-mt	logical-left-shift
cdr	eval	logical-not
ceiling	exit	logical-or
char<?	exp	logical-right-shift
char=?	expand-filename	logical-xor
char>?	expt	machine-id
char?	fasl-read	make-foreign
char-alphabetic?	file-directory?	make-string
char-downcase	file-exists?	make-vector
char->integer	file-modification-time	map
char-lower-case?	file-owner	max
char-numeric?	float	member
char-ready?	floor	memq
char-upcase	flush-output-port	memv
char-upper-case?	for-each	min
char-whitespace?	foreign?	mod
chdir	foreign-data	newline
clear-bit	foreign-id	not

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

nt?	set!	subvector-fill!
null?	set-bit	subvector->list
number?	set-car!	symbol?
number->string	set-cc	symbol-assigned?
oblist	set-cdr!	symbol-bound?
open-file	set-echo-ports!	symbol->string
open-input-file	sin	system
open-input-string	sqrt	system
open-output-file	stack-object	tan
open-output-string	stack-size	the-environment
or	string<?	time
output-port?	string=?	toggle-bit
pair?	string>?	trace-ignore
peek-char	string?	trace-off
port-echoing?	string-append	trace-on
port-name	string-ci<?	truncate
procedure?	string-ci=?	unix?
procedure-name	string-ci>?	valid-continuation
putenv	string-length	vector?
quotient	string->list	vector-length
read	string->number	vector-ref
read-char	string-ref	vector-set!
real?	string-set!	vms?
remainder	string->symbol	write
remove-file	substring	write-char
rename-file	substring-fill!	write-string
reverse	substring->list	

Fluent-Scheme Environment

The following list are all elements of the Fluent Scheme Environments. When it comes to functions the name is in brackets. Parameters are not specified. For list notes list only because the partial lists are very extensive and, otherwise, the value of the variable indicated, or n / a if the variable has no value.

Im folgenden sind alle Elemente des Fluent-Scheme Environments aufgelistet. Wenn es sich um Funktionen handelt, ist der Name in eckklammert – eventuell erforderliche Parameter sind nicht angegeben. Bei Listen ist nur list vermerkt, weil die Listen teilweise sehr umfangreich sind, ansonsten ist der Wert der Variablen angegeben, oder n/a falls die Variable keinen Wert hat.

solver-command-name fluent	(ti-set-turbo-topo)	(correct-turbo-defenition)
(client-file-version)	(gui-turbo-twod-contours)	(delete-turbo-topology)
(gui-get-selected-thread-ids)	(gui-turbo-avg-contours)	(define-turbo-topology)
(gui-show-partitions)	(gui-turbo-xyplots)	(setturbovar)
(gui-memory-usage)	(gui-turbo-report)	(getturbovar)
(grid-show)	(gui-set-topology)	(add-turbo-post-menu)
(rampant-menubar)	(ti-turbo-define)	(solve-controls-summary)
(gui-reload)	(ti-write-turbo-report)	(gui-solve-iterate)
(ti-avg-xy-plot)	(ti-compute-turbo-report)	(gui-solve-controls-mg)
(ti-2d-contour)	(write-turbo-data)	(gui-solve-controls-solution)
(ti-avg-contour)	(gui-turbo-define)	(order/scheme-name->type)

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(order/scheme-type->name)	(play-mpeg)	heatxc-groups list
order/schemes list	(cx-set-mpeg-compression)	heatxc-geom list
(name-list)	*mpeg-options*	(get-group-id)
(print-name->attribute-list)	*mpeg-qscale* 8	(get-hxg-id)
(print-name->pick-name)	*mpeg-bsearch* CROSS2	(new-hxg-id)
(print-name)	*mpeg-psearch* EXHAUSTIVE	(ti-get-model-input)
(get-eqn-units-patch)	*mpeg-range* 8	(ti-get-hxc-input)
(get-eqn-units-default)	*mpeg-pattern* IBBPBB	(ti-set-heatxc-model)
(get-eqn-var-default)	*mpeg-compression?* #f	(ti-set-hxc)
(get-eqn-var)	*mpeg-command*	(ti-hxc-report)
(set-eqn-var)	mpeg_encode	auto-set-porous? #f
(get-eqn-index)	(cx-animate)	(update-model-list)
(symbol->rpvar)	(cx-gui-animate)	(update-heatxc-list)
(inquire-equations)	(video-picture-summary)	(draw-all-macros)
(gui-solve-set-limits)	(video-summary)	(gui-heatxc-groups)
(gui-solve-set-ms)	(cx-video-use-preset)	(gui-heatxc-models)
(gui-patch)	(cx-video-show-picture)	(gui-heat-exchanger)
(gui-init-flow)	(cx-video-set-picture)	(pdf-init)
(gui-solar-calculator)	cx-video n/a	(inquire-species-names)
(gui-particle-summary)	(cx-video-show-options)	(inquire-n-species)
(models-summary)	(cx-video-set-options)	(surface-species-number)
(gui-dpm-sort)	(cx-video-close)	(surface-species-names)
(gui-models-dpm)	(cx-video-open)	(get-residual-norms-at)
(gui-models-viscous)	(cx-video-panel)	(residual-default-setting)
(gui-user-memory)	(cx-video-enable)	(ti-client-residuals-reset)
(gui-udf-on-demand)	*cx-video* #t	(client-residuals-reset)
(gui-udf-hooks)	cxvideo.provided #t	(client-support-residuals-reset?)
(gui-uds)	cxanim.provided #t	(residual-set)
(update-uds-domain-id-list)	(ti-del-hxg)	(residual-history)
(gui-models-soot)	(ti-set-hxg)	(unset-residual-norms!)
(gui-models-nox)	hxc-eff-vector #f	(set-residual-norms-by-max!)
(gui-vf-para)	(get-avail-zones)	(set-residual-norms-at!)
(gui-surface-glob)	(remove-frm-list)	(gui-monitor-residuals)
(gui-ray-trace)	(get-thread-id)	clres.provided #t
(gui-models-radiation)	(get-center)	(gui-solution-animation)
(set-radiation-model)	(set-porous-res)	(aniseq->path)
(gui-models-species)	(set-porous-dirs)	(aniseq->window)
(gui-models-multiphase)	(update-hxc-model)	(aniseq->storage)
(new-phase-name)	(initialize-hxc-model)	(aniseq->monitor)
(multiphase-model-changed)	(free-hxc-model)	(aniseq->display)
(gui-periodic-settings)	(heat-exchanger?)	(aniseq->name)
(gui-models-solidification)	(ti-get-res-hxc-input)	(ani-monitor-delete)
(gui-models-energy)	(get-hxc-opr)	(remove-ani-sequence)
(gui-operating-conditions)	(get-v)	(change-storage-type)
(gui-models-solver)	(set-drop-down-widgets)	(replace-sequence-by-name)
cx sweep.provided #t	(set-integer-widgets)	(change-sequence-window)
(cx-delete-keyframe)	(set-real-widgets)	(sequence-path-rename)
(cx-insert-keyframe)	(one-zone-group)	(sequence-rename)
(cx-display-frame)	(%init-hxc-model)	(add-animation-monitor)
keyframes list	(%free-hxc-model)	(ani-monitor-update)
(create-mpeg)	alstom #f	(set-animon-active?)
(mpeg-open)	heatxc-models list	(animon->active?)

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(animon->cmdstr)	(multiphase-vol-mon-compat)	multiphase-turbulence-menu list
(animon->when)	(gui-monitor-volume)	near-wall-treatment-menu list
(animon->freq)	(monitor-volume-init)	species-menu list
(animon->seqnum)	main-menu list	(ti-read-pdf-helper)
(animon->name)	grid-menu list	(enable-mixture-matl)
(build-ani-monitor-list-element)	turbo-menu list	solve-menu list
(remove-ani-xy-vars)	(turbo-set-current-topology)	initialize-menu list
xy-vars-list list	(turbo-avg-xy-plot)	initialize/compute-menu list
(set-xy-vars)	(turbo-2d-contours)	(set-domain-averaged-derived-flow-inits!)
(ani-save-xy-vars)	(turbo-avg-contours)	solve/set-menu list
(show-ani-monitors)	(write-turbo-report)	(set-eqn-vars)
(show-one-ani-monitor)	(compute-turbo-report)	(set-eqn/mg-controls)
(get-ani-monitors)	adapt-menu list	(set-eqn/scheme)
(ani-render-var-rename)	adapt/set-menu list	(set-eqn/solve)
(ani-show-thunk-titles)	reorder-menu list	(set-eqn/relax)
(ani-restore-thunk+title)	reorder-method-menu list	(set-eqn/default)
(ani-save-thunk+title)	(ti-reorder-using-cell-functions)	monitors-menu list
(ani-remove-thunk+title)	(ti-reorder-using-cell-distance)	solve/monitors-menu list
(ani-rename-monitor-thunk+title)	report-menu list	phase-menu list
(ani-restore-render-vars)	report/reference-menu list	pc-menu list
(ani-save-render-vars)	report/reference/compute-menu list	bc-menu list
(ani-monitor-active?)	display-menu list	modify-zones-menu list
(ani-monitor-name->seq)	plot-menu list	display/set-menu list
(ani-monitor-seq->name)	define-menu list	file-menu list
(ani-monitor-deactivate)	(ti-define-turbo-topology)	interpolate-menu list
(ani-monitor-activate)	ud-menu list	surface-cluster-menu list
(ani-monitor-change-freq)	(ti-udm)	file/export-menu list
(ani-monitor-rename)	(ti-execute-at-end)	file/import-menu list
(remove-ani-monitor)	(ti-udf-on-demand)	file/import/cgns-menu list
(add-ani-monitor-command)	(ti-ud-hooks)	file/import/partition-menu list
(run-ani-monitors)	profile-menu list	file/autosave-menu list
(animation-init)	operating-conditions-menu list	(allow-v2f-model)
(monitor-statistics-init)	(udf-models-changed)	allow-v2f-model? #f
solve/monitors/statistic-menu list	(set-uds-defaults)	cx-scene-menu list
(init-stats)	models-menu list	(ti-set-geometry)
(gui-monitor-statistics)	solver-menu list	(delete-cb)
(monitor-statistics)	radiation-menu list	(update-indices)
(gui-monitor-forces)	heat-exchanger-menu list	(ti-color-def)
(clear-monitor-forces)	(ti-uds)	(ti-transform)
(monitor-forces-init)	(update-pollutant-solve)	(ti-time-step)
(monitor-forces)	soot-menu list	(ti-path-attr)
(monitor-execute-at-end-transient)	nox-menu list	(ti-iso-sweep)
(monitor-execute-at-end)	(check-fuel-name-soot)	(ti-select-box-edge)
execute-string	(check-species-name2)	get-index #f
(gui-monitor-commands)	(check-species-name1)	ti-num-geoms 0
(monitor-command-init)	(check-fuel-name)	ti-selected-index list
(multiphase-surf-mon-compat)	s2s-menu list	ti-selected-geom list
(gui-monitor-surface)	dtrm-menu list	ti-selected-type list
(monitor-surface-init)	(multiphase-menu)	ti-selected-segment list
	turbulence-menu list	(cx-scene-update-geoms)
	turbulence-expert-menu list	(cx-scene-default-value)

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(scene-insert-order)	(repartition-remeshing-zones)	(delete-si-callback)
(cx-scene-insert-geoms)	(print-volume-skewness-limits-per-zone)	(create-si-callback)
(update-all-graphics)	(check-dynamic-mesh)	(copy-bc-callback)
(cx-scene-update-graphics)	(print-dynamic-forces-moments)	(change-zone-type-callback)
(recreate-geom?)	dynamic-mesh-menu list	remove-internal-layer-event 9
(restore-cx-globals)	dynamic-bc-menu list	insert-internal-layer-event 8
(save-cx-globals)	(eval-udf)	remove-layer-event 7
(close-gr-segments)	debug-dynamic-functions #f	insert-layer-event 6
(open-gr-segments)	(clear-dynamic-functions)	change-time-step-event 5
(cx-scene-draw-cmap)	(update-dynamic-functions)	change-motion-attr-event 4
(redisplay-all)	(cancel-dynamic-function)	delete-sliding-interface-event 3
(cx-scene-set-iso-surf)	(register-dynamic-function)	create-sliding-interface-event 2
(cx-get-scene-update)	(ti-position-starting-mesh)	copy-zone-event 1
(cx-set-scene-update)	(get-remesh-cell-threads)	change-zone-event 0
(cx-scene-list-geometry)	(update-solver-thread-names)	(lift->angle)
(scene-get-string)	(set-dt-mask)	(crank-angle->time)
(cx-show-user-option)	(mask-names->mask)	(crank-angle->absolute-time)
(cx-transform-highlight)	(mask->mask-names)	(time->crank-angle)
(cx-draw-bbox)	cell-element-type-alist list	(nth-ic-cycle)
(cx-flush-bbox)	(ti-modify-lift)	(time->absolute-crank-angle)
(cx-scene-show-bbox)	(ti-print-plot-lift)	(fmod)
(cx-set-vv-attr)	(plot-valve-lift)	(gui-dynamic-zone-preview)
(cx-set-profile-attr)	(print-valve-lift)	(display-surfaces)
(cx-set-dpm-attr)	(ti-delete-internal-layer)	(advance-mesh)
(cx-set-path-attr)	(ti-insert-internal-layer)	(auto-hardcopy)
(cx-set-contour-attr)	(ti-remove-layer)	(animate-motion)
(get-viz-iso-surf-id)	(ti-insert-layer)	(gui-motion-preview)
(iso-surface-ancestor)	(remove-layer)	(preview-motion)
(derived-from-iso-surface)	(insert-layer)	(update-all-geom-positions)
(show-surface-units)	(delete-internal-layer)	(update-one-geom-position)
(show-surface-quantity)	(insert-internal-layer)	(v3-rsub)
(show-surface-type)	(cleanup-thread-list)	dz-ani-storage-name
cx-scene-panel-present #f	(update-in-cylinder-monitors)	dynamesh_preview
(cx-gui-scene)	(monitor-crank-angle)	dz-ani-storage-type 2
(cx-gui-bbox-frame)	(gui-ic-event-playback)	dz-ani-sequence -1
(insert-projections)	(ic-event-playback)	dz-animate? #f
(inc-geoms)	(gui-ic-events)	dz-nstep 1
(insert-planes)	(ic-event-hook)	dz-display-frequency 1
(add-delta)	(handle-ic-event)	dz-display? #t
(cx-display-bnd-frame)	(inside-range)	dz-hardcopy? #f
cx-frame-growth-factor 0.01	(shift-down)	(contour-node-displacement)
cx-frame-domain? #t	(shift-up)	(compute-cg-from-profile)
ctxicks.provided #t	angle-tol 1e-05	(zone-selection)
cxscene.provided #t	event-callback-alist list	(disable-items)
ti-non-reflecting-menu list	(remove-internal-layer-callback)	(enable-items)
(remesh-local-prism-faces)	(insert-internal-layer-callback)	(hide-items)
(print-remesh-cell-marks)	(remove-boundary-layer-callback)	(show-items)
(draw-remesh-cell-marks)	(insert-boundary-layer-callback)	(gui-dynamic-zones)
(mark-remesh-cells)	(change-motion-attr-callback)	(gui-models-moving-grid)
(refine-coarsen-on-skewness-size)	(change-time-step-callback)	(ti-list-dynamic-threads)
(remesh-local-cells)		(ti-reset-dynamic-thread)
		(delete-adjacent-dz)

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(delete-dynamic-thread)	(dpm-parallel-message-passing)	(acoustic-reader-gui)
(ti-set-dynamic-thread)	(ti-particle-tracks)	(receivers-gui)
(ti-enable-dynamic-mesh)	(pathline-summary)	(acoustics-write-sound-gui)
(enable-dynamic-mesh)	(dpm-summary)	(thread-acoustic-gui)
(change-dz-attr)	(dpm-iteration)	(append-ac-file)
(new-copy-of-dz)	(pick-particle-cell-function)	(write-next-acoustic-timestep)
(set-dz-attr)	(inquire-particle-cell-functions)	(open-acoustic-file-for-write)
(get-dz-attr)	(dpm-display-path-lines)	(acoustics-trans-freq-proc)
(get-dynamic-thread)	(injection-types)	(acoustics-file-write-and-compute)
deforming-motion 3	law-list list	(append-file-name-to-index-file)
user-defined-motion 2	property-list list	(get-receiver)
solid-body-motion 1	(pick-law)	exporting-data-only #f
no-motion 0	(custom-laws?)	(acoustics-model-changed)
(motion-type-name->id)	(ti-recover-law)	(allow-acoustics-model)
(motion-type-id->name)	(ti-convert-law)	allow-acoustics-model? #t
(new-dz)	(ti-dpm-law-options)	(ti-crevice-menu)
(read-reference-vol)	(dpm-law-options)	(ti-make-crevice-threads)
(write-reference-vol)	(dpm-default-laws)	(new-crevice-case)
(set-remesh-repartition-thresholds)	(dpm-material-type)	(crevice-summary)
(set-sizing-function-defaults)	(combusting-not-multi-surface?)	(valid-flmon-chkpnt-dir?)
(set-sizing-function-dimension-defaults)	(multi-surface?)	flmon-chkpnt-dir list
(update-dynamesh-hooks)	(comb-method)	(flmon-init)
(allow-dynamic-mesh)	(pick-particle-type)	flmon-running? #f
allow-dynamic-mesh? #t	(pick-injection-type)	(valid-sge-chkpnt-dir?)
preview-auto-save #f	(pick-dpm-material)	sge-chkpnt-dir list
execute-ic-event #t	(pick-stream)	(sge-init)
debug-ic-event #f	(pick-species)	(lsf-spawn)
(ti-export-event)	(pick-generic)	(valid-lsb-chkpnt-dir?)
(ti-import-event)	(dpm-is-cloud-on?)	lsb-chkpnt-dir list
(free-dynamic-mesh)	gui-law-define n/a	(lsf-init)
(dynamic-thread-vars-compat)	(pick-injection)	(monitor-send-exit-sig)
(dynamic-thread-list)	(inquire-injection-names)	(monitor-new-procs)
(unthread-dynamic-thread)	dpm-menu list	(contact-monitor)
(thread-dynamic-thread)	dpm-injections-menu list	(monitor-config-file)
(download-dynamic-threads)	(dpm-bcs-available?)	(write-kill-script)
(create-case-dynamic-threads)	(dpm-material-types)	(exit-restart-file)
(gui-s2s)	(activate-injection-surfaces)	(restart-commands)
(gui-dtrm)	(dpm-change-material-name)	(checkpoint)
(display-sample-points)	(dpm-used-material-names)	(create-checkpoint-fnbase)
(gui-samples-manage)	gui-dpm-display-particle-traces n/a	(create-checkpoint-filename)
(pick-sample)	(get-all-dpm-material-names)	(remove-checkpoint-files)
(sample-name)	(download-injections)	(set-checkpoint-variable)
(delete-sample)	(create-case-injections)	(chkpnt-dir)
(list-sample-fields)	(gui-manage-injections)	(valid-chkpnt-dir?)
(list-samples)	(reset-injections)	(valid-exit-file?)
(plot-sample-histograms)	(free-injections)	(valid-check-file?)
(update-sample-list-cache)	acoustics-menu list	(create-exit-file)
(sample-list)	(ti-receivers-specification)	(create-check-file)
(ti-dpm-sample-report)	(ti-source-specification)	last-data-filename #f
(dpm-parallel-shared-memory)	(ti-read-and-compute)	last-case-filename #f
	(gui-models-acoustics)	save-rc-filename #f

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

checkpoint/exit-filename	(list->user@host-dot)	(species-mass-flow)
/tmp/exitfluent	(user@host->list)	(mass-flow)
checkpoint/check-filename	(create-hosts-db)	(cur-to-si-unit)
/tmp/checkfluent	(delete-from-available-hosts)	(si-to-cur-unit)
checkpointed? #f	list-compute-nodes list	(ti-set-reference-var)
(benchmark)	list-of-available-hosts list	(advance-oned-solution)
(debug-node)	list-hosts list	(update-wave-bcs)
(debug-all)	(mkill?)	(start-oned-library)
(debug-client)	(valid-partition-id?)	(free-oned-library)
(attach-debugger)	(ti-translate-grid)	(draw-oned-cell)
(attach-ibm-dbx)	(ti-scale-grid)	library-list list
(attach-sgi-dbx)	(translate-grid)	(oned-library-gui)
(attach-sun-dbx)	(gui-translate-grid)	(display-profile-points)
(attach-ladbug)	(scale-grid)	(ti-write-profiles)
(attach-gdb)	(gui-scale-grid)	(gui-write-profiles)
(repartition-by-zone)	cxgrid.provided #t	(gui-profiles-manage)
(gui-load-balance)	(ti-summary)	(gui-profile-orient)
(gui-show-virtual-machine)	(gui-summary)	(axial-profile-to-xy)
(gui-network-configure)	(gui-volume-integrals)	(radial-profile-to-xy)
(gui-hosts-data-base)	(ti-cell-thread-integral)	(ti- conserve-total-enthalpy)
(gui-auto-partition-grid)	(ti-cell-thread-volume)	(ti-create-mixing-plane)
(gui-partition-grid)	(gui-reference-values)	(ti- conserve-swirl)
parallel-menu list	(ti-plot-histogram)	(ti-set-pressure-level)
partition-menu list	(ti-print-histogram)	ti-mixing-plane-menu list
partition/auto-menu list	(gui-histogram)	(gui-mixing-plane)
(ti-partition-auto)	(plot-histogram)	(initialize-mixing-plane-profiles)
partition/set-menu list	(print-histogram)	(update-mixing-planes)
(ti-partition)	(client-histogram-bins)	(write-profiles)
(pick-partition-function)	(client-histogram-max)	(pick-profile)
parallel/network-menu list	(client-histogram-min)	(profile-name)
parallel/timer-menu list	*cx-histogram-location* cells	(update-profiles)
parallel/set-menu list	cxreports.provided #t	(delete-profile)
(gui-reorder-domain)	(gui-wall-reports)	(list-profile-fields)
(ok-to-invalidate-case?)	(print-wall-moments)	(list-profiles)
(spawn-from-file)	(print-wall-forces)	(update-user-function-list-cache)
(spawn-from-list)	(wall-moments)	(user-function-list)
(spawn-compute-node)	(wall-forces)	(update-profile-list-cache)
(add-to-available-hosts)	(viscous-moment)	(profile-list)
(first-word)	(viscous-force)	ti-real-gas-init list
(make-hosts-db-available)	(pressure-moment)	(ti-ud-real-gas)
(read-partition-id)	(pressure-force)	(open-udrg-library)
(read-partition-id-list)	(iprint-wall-moments)	(ti-nist-real-gas)
(disable-load-balance-after-adaption)	(iprint-wall-forces)	(ti-real-gas-dataname)
(enable-load-balance-after-adaption)	(gui-thread-reports)	(open-rgas-library)
(check-partition-encapsulation)	(thread-integrals)	(default-property-constant)
use-default-auto-partition? #t	(sort-threads-by-name)	(update-species-variable-lengths)
case-file-partition-pretest? n/a	(shell-heat-transfer)	(properties-changed)
case-file-partition-method n/a	(rad-heat-transfer)	(species-changed)
(ti-migrate-marked-cells)	(heat-transfer)	(client-list-reactions)
(migrate-marked-cells)	phase-mass-flow n/a	(client-material-name-changed)
	phase-volume-flow n/a	(client-set-material!)
	(uds-flow)	

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(client-used-material-names)	(ti-change-material)	(free-bridge-nodes)
(client-material-types)	(gui-materials-manage)	(fill-bridge-nodes)
(client-property-names)	(list-database-materials)	(gui-smooth-grid)
(client-property-list)	(database-property-methods)	(ti-make-hanging-interface)
(ti-import-chemkin)	(get-database-material-copy)	(ti-swap-mesh-faces)
materials-menu list	(get-database-material-copy-by-	(swap-mesh-faces)
(ti-copy-material-by-formula)	formula)	(ti-smooth-mesh)
(ti-copy-material-by-name)	(get-database-material-by-	(smooth-mesh)
materials/database-menu list	formula)	(gui-yplus-adapt)
(import-chemkin-mechanism)	(get-database-material)	(gui-volume-adapt)
(load-pdf-species)	(property-units)	(gui-region-adapt)
(property-method-assq)	(property-name)	(gui-isovalue-adapt)
(property-method-name-prmx)	(inquire-database-material-	(gui-gradient-adapt)
(property-method-name)	names)	(gui-boundary-adapt)
(default-property-data)	(inquire-material-names-all)	(gui-manage-mark)
(add-mixture-name)	(inquire-material-names)	(gui-display-mark-options)
(qlist-)	(get-default-material-name)	(gui-adapt-controls)
(total-reaction-list)	(get-names-of-type)	(gui-display-contours-or-error)
(mlist-)	(get-first-material-name)	(create-new-interior-threads)
(alist-)	(pick-material-type)	(set-register-ncrsn!)
(material-site-nspecies)	(pick-database-material-by-	(set-register-nrefn!)
(material-site-species-names)	formula)	(set-register-cbit!)
(material-surface-nspecies)	(pick-database-material-by-	(set-register-rbit!)
(material-surface-species-	name)	(set-register-type!)
names)	(pick-database-material)	(set-register-name!)
(material-volumetric-nspecies)	(pick-material-all-with-prompt)	(set-register-id!)
(material-volumetric-species-	(pick-material-with-prompt)	(register-ncrsn)
names)	(pick-material-all)	(register-nrefn)
(material-nspecies)	(pick-material)	(register-cbit)
(material-species-names)	(list-database-properties)	(register-rbit)
(material-species-names-pair)	(list-properties)	(register-type)
(material-profiles)	(list-materials)	(register-name)
(reaction-list-for-material)	(get-material-copy)	(register-id)
(material-prop)	(get-materials-of-type)	(mark-percent-of-ncells)
(material-types)	(get-material)	(ti-mark-percent-of-ncells)
(material-mixture-pair)	(set-material-properties!)	mask-register #f
(material-type)	(set-material!)	refn-register #f
(material-formula)	(set-all-materials!)	(ti-free-parents)
(mixture-material-name)	(get-n-materials)	(refine-mesh)
(material-name)	(get-all-materials)	(ti-refine-mesh)
(material-in-use-by-mixture)	(delete-material)	(adapt-mesh)
(dpm-check-material-in-use)	(update-material-properties)	(ti-adapt-mesh)
max-material-name-len 25	(update-case-materials)	(register-invert)
all-material-types list	(create-case-materials)	(ti-register-invert)
(materials-summary)	(free-materials)	(ti-mask-invert)
(client-property-methods)	(create-mixture-material)	(swap-refn-crsn)
(reset-prop-methods-cache)	(copy-database-material)	(ti-swap-refn-crsn)
(update-case-material-	(create-material)	(draw-marked-cells)
properties)	(upload-materials)	(draw-node-flags)
(ti-delete-material)	(download-materials)	(ti-draw-marked-cells)
ti-create-material n/a	cxprop.provided #t	(mark-inout-iso-range)
(ti-copy-material)	(draw-bridge-nodes)	(ti-mark-inout-iso-range)

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(mark-inout-shape)	(ti-limit-marked-cells)	(ti-surface-mass-average)
(ti-mark-inout-shape)	(ti-count-marked-cells)	(ti-surface-flow-rate)
(ti-shape-define)	(replace-register)	(ti-surface-mass-flow-rate)
(mouse-shape-define)	(free-registers)	(ti-surf-vertex-max)
(auto-refine-level)	(ti-free-registers)	(ti-surf-vertex-min)
(mark-with-refine-level)	(destroy-register)	(ti-surf-facet-max)
(ti-mark-with-refine-level)	(ti-destroy-register)	(ti-surf-facet-min)
(mark-with-volume-change)	(get-def-register)	(ti-surf-min-max)
(ti-mark-with-volume-change)	(create-copy-of-register)	(ti-surface-integral)
(mark-boundary-cells-per-thread)	(create-register)	(ti-surface-average)
(ti-mark-boundary-cells-per-thread)	adapt-env list	(ti-surface-area)
(mark-with-volume)	(surface-ids->name-pair)	(print-one)
(ti-mark-with-volume)	(thread-surface)	(unit-label)
(mark-with-boundary-volume)	(ithread-surface)	(unit-value)
(ti-mark-with-boundary-volume)	(thread-coordinates)	(print-header)
(mark-with-yplus-per-thread)	(thread-values)	(gui-fill-face-zone-values)
(ti-mark-with-yplus-per-thread)	(inquire-zone-names)	(cx-fill-face-zone-values)
(mark-with-ystar-per-thread)	(zone-name->id)	(surface-vertex-max)
(ti-mark-with-ystar-per-thread)	(zone-id->name)	(surface-vertex-min)
(mark-with-gradients)	(zone-var)	(surface-vertex-average)
(ti-mark-with-gradients)	(zone-type)	(surface-facet-max)
(ti-adapt-to-boundary-cells)	(zone-name)	(surface-facet-min)
(ti-adapt-to-refine-level)	(read-zone-id-list)	(surface-facet-average)
(ti-adapt-to-volume-change)	(read-zone-id)	(surface-sum '(7) "velocity-magnitude")
(ti-adapt-to-ystar-per-thread)	(get-zone)	(surface-massavg)
(ti-adapt-to-ystar)	(zone-id)	(surface-mass-average)
(ti-adapt-to-yplus-per-thread)	cx-surface-menu list	(surface-mass-flow-rate)
(ti-adapt-to-yplus)	(ti-surface-projected-area)	(surface-flow-rate)
(mark-with-gradients)	(pick-surface-group)	(surface-integral)
(ti-adapt-to-default-register)	(read-new-surface-id.name)	(surface-average)
(adapt-to-register)	(iline-surface)	(surface-area '(7))
(ti-adapt-to-register)	(irename-surface)	(surface-integrate)
(list-registers)	(idelete-surface)	(cx-delete-srf-ref-in-grp)
(combine-list-of-registers)	(icell-surface)	(surface-ids->surface-groups)
(combine-registers)	(ipoint-surface)	(surface-id->surface-group)
(ti-combine-registers)	(ipoint-array)	(cx-reset-surface-groups)
(get-mask-registers)	(iiso-clip)	(cx-init-surface-groups)
(get-refn-registers)	(iiso-surface)	(make-surface-groups)
(get-all-registers)	(izone-surface)	(inquire-surface-line-names)
(get-register)	(ipartition-surface)	(inquire-surface-plane-names)
(ti-read-mask-register)	(isphere-slice)	(inquire-surface-group-names-of-type)
(ti-read-refn-register)	(iplane-slice)	(inquire-surface-group-names)
(ti-read-register-list)	(irake-surface)	(grp->srf)
(ti-read-register)	(isurface-cells)	(cx-rename-srf-group)
(toggle-register-type)	(isurface-grid)	(cx-get-group-srfs)
(ti-toggle-register-type)	(gui-surface-projected-area)	(srf-grp?)
(fill-crsn-register)	(default-minimum-feature-size)	(cx-delete-group)
(ti-fill-crsn-register)	(gui-surface-integrals)	(cx-add-new-srf-group)
(limit-marked-cells)	(ti-surface-vertex-average)	(cx-ungroup)
	(ti-surface-facet-average)	(remove-from-grp-list)
	(ti-surface-sum)	
	(ti-surface-massavg)	

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(cx-group-grps)	*cx-surface-list-width* 25	(cutting-plane-off)
(flatten-surface-groups)	(read-surface-id-list)	(create-cutting-plane)
(surface-values)	(read-surface-id)	(cutting-plane-hook)
(surface-coordinates)	(read-surface-list)	(cx-activate-plane-tool)
(make-pairs)	(read-surface)	cxplane.provided #t
(pair-coords)	(cx-get-surface-ids)	(ti-custom-field-function/load)
(cx-surface-face-list)	(surface-name/id?)	(ti-custom-field-function/save)
(cx-surface-uniq-node-coords)	(valid-surf-name?)	(cf-code)
(cx-surface-uniq-node-values)	(rake-surface)	(pp-cfd)
(surface-velocity-vectors)	(mrake-surface)	(cx-get-obj-desc-attr)
(cx-surface-coordinates)	(line-surface)	(cx-get-obj-attr)
(cx-surface-values)	(mline-surface)	(cx-get-obj-id)
(zone-coordinates)	(plane-surface)	(cx-store-obj-all-attr)
(zone-values)	(mplane-surface)	(cx-store-obj-desc-attr)
(apply-slice)	(quadric-surface)	(cx-store-obj-attr)
(surface-facets)	(sphere-slice)	(cx-get-obj-by-attr)
(rename-surface)	(ibounded-plane)	(cx-get-obj)
(temp-surface?)	(iplane-surface-align)	(cx-delete-obj)
(surface?)	(point-normal-surface)	(cx-add-obj)
(list-surfaces)	(plane-slice)	(cx-new-obj-id)
(surface-area-vectors)	(point-array)	(cx-cf-name->id)
(surface-name)	(sphere-coeff)	(cx-cf-id-attr)
(get-surface)	(plane-coeff)	(cx-inquire-cf-ids)
(new-surface-id)	(iso-clip)	(cx-inquire-user-def-cf-names)
(get-mouse-point)	(iso-clip-new)	(cx-get-cf-desc-attr)
(fill-cx-tmp-array)	(partition-surface)	(cx-get-cf-attr)
(gui-surfaces-creation-failure)	(sample-plane-points)	(cx-set-cf-attr)
(gui-add-named-surface)	(planar-point-surface)	(cx-get-cf-by-attr)
(gui-transform-surface)	(point-surface)	(cx-rename-cf)
(cx-rotate-3d)	(transform-surface)	(cx-delete-cf)
(cx-scale-mat)	(cell-surface)	(cx-get-cf)
(gui-iso-clip)	(zone-surface)	(cx-add-cf)
(gui-iso-surface)	(iso-surface)	(cx-new-cf-id)
(gui-quadric-surface)	(surface-append!)	(cx-initialize-cell-functions-
(gui-plane-surface)	(delete-surfaces)	client)
(cx-show-plane-tool-normals)	(surface-grid)	(cx-eval-cf)
(gui-line/rake-surface)	(suspend-surfaces)	(code-gen)
(cx-show-line-tool-normals)	(free-surfaces)	(d/dz)
(gui-point-surface)	(iso-srf-chk)	(d/dy)
(gui-partition-surface)	(cx-restart-fast-iso)	(d/dx)
(gui-zone-surface)	(cx-end-fast-iso)	(gui-manage-cf)
(gui-manage-surfaces)	(cx-start-fast-iso)	(custom-field-function/load)
(cx-remove-tmp-geom)	(cx-destroy-surface-all)	(custom-field-function/save)
(cx-display-tmp-surf)	(cx-destroy-surface)	(custom-field-function/define)
(cx-surface-get-min-max)	(cx-delete-zone-surface)	(err-reduction-fail)
(cx-surface-fill-temp)	(surface-id->zone-id)	(match-all-productions)
(rem-quote-sym)	(zone-id->surface-id)	(match-production)
(cx-get-surf-def-attr)	(cx-create-boundary-zone-	(push-sym)
(cx-set-surf-def-attr)	surfaces)	(pop-sym-till-less-prec)
(cx-get-surf-def-attr-pos)	(create-zone-surface)	(inc-parse)
cx-surf-interp-attr-list list	(add-zone-surface-defs)	(token-value)
(cx-copy-surface)	(client-inquire-fast-iso)	(token-syn-cat)

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(top-terminal)	(dot)	(cx-store-surface-desc-attr)
(set-precedence!)	(row-mat)	(cx-store-surface-attr)
(precedence)	(mm-aux)	(cx-get-surface-attr)
(sym-value)	(aref)	(cx-get-surface-desc-attr)
(end-parser)	(mat-mult)	(surface-id)
(init-parser)	(cx-identity-mat)	(cx-active-surface)
parse-stack list	(cx-translate-mat)	(cx-set-surface)
(value-prod)	(cx-rotate-x)	(cx-get-surface)
production-list list	(cx-rotate-y)	(cx-set-surface-lists)
(shift-error-check)	(cx-rotate-z)	(set-surface-version)
(err-table-ref)	(max-list)	(surface-version)
(parse-table-ref)	(min-list)	(cx-save-surface-lists)
col-entries list	(transpose)	(surf-set-list-size)
parse-table list	(transform)	(surf-list-bnds-chk)
(parse)	(cx-get-trans-pts-min-max)	surfaces-groups list
(lex)	(cx-update-surface-attr)	cx-max-surf-id-ref 21
(init-number)	(cx-update-all-surface-attr)	cx-surface-list #f
(init-lexer)	(list-union)	cx-temp-surfaces? #f
(init-parser/lexer)	(cx-ancestor-surfaces-id)	cx-temp-surface-list list
prev-token #f	(cx-ancestor-surfaces-id-list)	first-virtual-id 4196
prev-number? #f	(cx-purge-surface-def-list)	*cx-max-surface-num* 4096
cur-number-str	(iso-func)	*cx-fast-iso-info* #f
(gui-user-def-cf)	(cx-create-surface-from-def)	*cx-big-neg* -1e+20
(but-name->display)	(cx-generate-susp-surface-defs)	*cx-big-pos* 1e+20
(bin-op?)	(cx-add-surface-def)	surf.provided #t
(un-op?)	(cx-get-def-coarse-surface)	(set-cx-field-render-vars)
(trig-op?)	(virt2real)	(fill-face-thread-values)
trig-ops list	(vt2rl)	(fill-face-values)
un-ops list	(real2virt)	(fill-cell-values)
bin-ops list	(rl2vt)	(fill-node-values)
calc-display-map list	(cx-delete-virtual-id)	(inquire-cell-functions-
(var-name->display)	(cx-get-virtual-index)	sectioned)
calc-display list	(cx-delete-map-entry)	(inquire-cell-functions)
calc-inp-list list	(cx-list-surfaces)	(%client-inquire-cell-vector-
cf-str	(cx-surface-area-vectors)	functions)
cx-cell-function-length 75	(surface-id->name)	(%client-inquire-cell-functions-
cx-max-cf-name 25	(surface-ids->names)	sectioned)
(cx-field-rename)	(surface-name->id)	(%client-inquire-cell-functions)
(cx-field-eval)	(inquire-point-surface-names)	(client-support-symmetry?)
(cx-field-define)	(inquire-surface-names)	(display-grid-partition-boundary)
cxcf.provided #t	(inquire-surface-ids)	(client-draw-grid-partitions)
cxsurf.provided #t	(cx-suspend-all-surfaces)	(display-grid-outline)
(surf-inits!)	(cx-update-surface)	(display-grid)
(pp-list)	(cx-create-surface)	(idraw-thread-grid)
(v3-interpolate)	(cx-rename-surface)	(grid-internal)
(v3-unit)	(cx-delete-surface)	(grid-outline)
(v3-magnitude)	(cx-add-surface)	(thread-grid)
(v3-cross)	(cx-new-temp-surface-index)	(gui-animation-control)
(v3-dot)	(set-next-surface-index)	(animation-name-list)
(v3->z)	(new-surface-index)	(cxg-ani-hardcopy-frames-cb)
(v3->y)	(cx-active-surface-ids)	(create-mpeg-animation)
(v3->x)	(cx-store-surface-all-attr)	

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(cx-set-hardcopy-options-for- mpeg)	(cx-xy-plot-data)	(ti-write-es-gold)
(get-node-field-list-seq)	(cx-solution-plot)	(ti-write-es)
(get-node-field-list-win)	(cx-surface-plot)	(ti-write-fv-data)
(show-node-field-list)	(cx-zone-plot)	(ti-write-fv)
(cxg-copy-win-field-to-seq-field)	(surface-positions)	(ti-write-avs)
(cxg-restore-seq-node-field)	(gui-xy-plot-curves)	(ti-write-fv-uns)
(cxg-update-active-window- node-field)	(ti-set-xy/scale/fmt)	(ti-write-tecplot)
(cxg-save-node-field)	(gui-xy-plot-axes)	(write-radtherm)
(cxg-ani-last-hardcopy-frame- list)	(cx-xy-plot-files)	(write-patran)
(cxg-ani-create-hardcopy- frames)	(xy-plot-file)	(ti-write-cgns)
(cxg-ani-hardcopy-callback)	(xy-read-file)	(write-cgns)
(cxg-ani-hardcopy-filename)	(xy-read-port)	(write-gambit)
(cxg-ani-replay)	my-multi? #f	(write-flpost)
(cxg-update+xy-animation)	(xy-read-columns)	(write-fast)
(cxg-ani-xy-plot)	(xy-read-curves)	(write-engold-gui)
(cxg-update+snap-animation)	(xy-read-particle)	(write-engold-ascii)
(cxg-snap-animation)	(xy-build-particle-curves)	(write-engold-binary)
(cxg-ani-check-path)	(xy-read-particle-header)	(write-engold)
(cxg-ani-get-seq-basename)	(xy-plot-list)	(write-es-gold-transient)
(seqlist->winid)	(end-plot)	(write-es-transient)
(seqlist->frames)	(start-title-plot)	(write-es-gold)
(seqlist->storetype)	*cx-xy-multiple-files* #t	(write-es)
(seqlist->name)	cxyy.provided #t	(write-fv-data)
(ti-fft-plot-file)	cxganim.provided #t	(write-fv)
(ti-xy-plot-radial-band-averages)	(iread-bc)	(write-fv-uns)
(radial-band-average)	(iwrite-bc)	(ti-write-dx)
(ti-xy-plot-axial-band-averages)	(downcase)	(write-dx)
(axial-band-average)	(process-thread)	(write-avs)
(band-average)	(set-dv-variables)	(write-tecplot)
(scalar-bands)	(set-rp-variables)	(write-icepak-results)
(band-clip)	(set-bc)	(ti-write-icemcfd)
(ti-xy-plot)	(list-bc)	(write-icemcfd)
(ti-xy-plot-zone/surface)	(read-bc)	(ti-write-flux-profile)
(ti-xy-set-surface-scale)	(write-bc)	(write-radiation-export)
plot/set-menu list	(has-wild-card?)	(ti-write-radiation)
(ti-surface-plot)	(get-thread-list)	(enable-radtherm)
(ti-zone-plot)	(gui-autosave-files)	(ti-write-radtherm)
(ti-solution-plot)	(autosave-case-data)	(ti-write-ansys)
(ti-xy-set-scale)	(write-transient-ensight-explicit)	(write-ansys)
(ti-xy-plot-files)	(write-transient-ensight-auto- append)	(ti-write-ascii)
(ti-xy-plot-file)	(write-transient-ensight-case)	(write-ascii)
(xy-plot-zones+surfaces)	(append-transient-export)	(ti-write-ideas)
(cx-xy-plot-buffer-data)	(write-transient-ensight-scalar)	(write-ideas)
(gui-xy-plot-ffts)	(write-transient-ensight-vel)	(ti-write-abaqus)
(gui-xy-plot-files)	(write-transient-ensight-geo)	(write-abaqus)
(gui-xy-plot-zone/surface)	(autosave-freq-proc)	(ti-write-nastran)
(cx-xy-plot-to-file)	(gui-write-export)	(write-nastran)
(cx-xy-plot-to-port)	(ti-write-gambit)	(ti-write-patran-cell- temperature)
	(ti-write-fast)	(ti-write-patran-nodal-results)
	(ti-write-es-transient)	(ti-write-patran-neutral-file)
	(ti-write-es-gold-transient)	(write-patran-nodal-results)

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(write-patran-result-template)	(import-ansys)	(gui-import-cgns-data)
(write-patran-neutral-file)	(gui-import-fluent4-case)	(ti-read-cgns-data)
(write-patran-cell-temperature)	(ti-import-fluent4-case)	(cgns-data-read)
enable-radtherm-export? #t	(import-fluent4-case)	(update-solver-threads)
(ti-write-fast-solution)	(gui-import-metis-zone-case)	(ti-write-fan-profile)
(ti-write-fast-scalar)	(gui-import-metis-case)	(write-fan-profile)
(ti-write-fast-velocity)	(ti-import-metis-zone-case)	(gui-write-hosts)
(ti-write-fast-grid)	(ti-import-metis-case)	(ti-write-hosts)
(write-fast-solution)	(import-metis-zone-case)	(write-hosts)
(write-fast-scalar)	(import-metis-case)	(gui-read-hosts)
(write-fast-velocity)	(client-write-data)	(ti-read-hosts)
(write-fast-grid)	(client-read-data)	(read-hosts)
fast-binary-files? #f	(client-append-data)	(read-isat-table)
(ti-write-mpgs-scalar)	(client-write-case)	(ti-read-isat-table)
(ti-write-mpgs-velocity)	(client-read-case)	(gui-read-isat-table)
(ti-write-mpgs-geometry)	(client-read-zone)	(write-isat-table)
(write-mpgs-gold-scalar)	(reading-case-file)	(ti-write-isat-table)
(write-mpgs-scalar)	(force-metis-method-	(gui-write-isat-table)
(write-mpgs-gold-velocity)	compatibility)	(gui-write-surface-globs)
(write-mpgs-velocity)	(rpsetvar-to-default)	(ti-write-surface-globs)
(write-mpgs-gold-geometry)	(particle-history-open?)	(write-surface-globs)
(write-mpgs-geometry)	(stop-particle-history)	(gui-write-boundary-grid)
(write-export)	start-particle-history-write n/a	(ti-write-boundary-grid)
(ti-open-oned-library)	(start-particle-history)	(write-boundary-grid)
(ti-open-udf-library)	(gui-start-particle-history)	(gui-reread-grid)
(ti-udf-compile)	(ti-start-particle-history)	(ti-reread-grid)
(gui-udf-compile)	(write-solar-pos)	(reread-grid)
(open-isat-library)	(gui-write-sglobs)	(gui-read-sample)
(open-udf-library)	(ti-write-sglobs)	(read-sample)
(load-udf-scm-file)	(write-sglobs)	(gui-write-existing-profile)
(udf-compile)	(ti-write-viewfac)	(ti-write-existing-profile)
(compare-case/solver)	(write-viewfac)	(write-existing-profile)
(gui-import-patran)	(gui-read-sglobs-vf)	(gui-read-profile)
(ti-import-patran)	(ti-read-sglobs-vf)	(ti-read-transient-table)
(import-patran)	(read-sglobs-vf)	(ti-read-profile)
(gui-import-nastran)	(gui-read-sglobs)	(read-transient-table)
(ti-import-nastran)	(ti-read-sglobs)	(read-profile)
(import-nastran)	(read-sglobs)	export-file-types n/a
(gui-import-ideas-universal)	(gui-write-rays)	(write-transient-ensight)
(ti-import-ideas-universal)	(ti-write-rays)	pload #f
(import-ideas-universal)	(write-rays)	loc #f
(gui-import-gambit)	(gui-read-rays)	filename n/a
(ti-import-gambit)	(ti-read-rays)	frequency-entry n/a
(import-gambit)	(read-rays)	htc-flux? #t
(gui-import-fidap)	(gui-read-pdf)	htc-wall? #f
(ti-import-fidap)	(ti-read-pdf)	htc-wall #f
(import-fidap)	(read-pdf)	(ensight-frequency)
(gui-import-cgns)	(gui-interp-data)	transient-on n/a
(ti-import-cgns)	(ti-write-interp-data)	transient? n/a
(import-cgns)	(ti-read-interp-data)	htc n/a
(gui-import-ansys)	(interp-data)	loads n/a
(ti-import-ansys)	(gui-import-chemkin)	transient n/a

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

location n/a	clrelease.provided #t	(create-case-shells)
delimiter n/a	*client-library-run-time-release*	si-menu list
surfaces n/a	3	(ti-make-periodic-interface)
binary? #t	clfiles.provided #t	(ti-draw-zones)
node n/a	(gui-open-udf-library)	(ti-list-sliding-interfaces)
space n/a	(ti-compile-now)	(list-sliding-interface)
cell-centered n/a	(bc-summary)	(ti-delete-all-si)
comma n/a	(domainvars-rpvars-compat)	(delete-all-sliding-interfaces)
(check-data)	(initialize-domain-menus)	(ti-delete-si)
(check-grid)	(ti-set-type-domain-vars)	(ti-create-si)
cl-file-package list	(read-domain)	(si-thread?)
(suffix-expand-filename)	(gui-domains-manage)	(gui-grid-interfaces)
(gui-import-data)	(clear-and-hide-panel)	(inquire-si-threads)
(ti-import-data)	(download-domains)	(sliding-interface-thread?)
(gui-import-case)	(get-all-domain-vars)	(interface-in-use?)
(ti-import-case)	(create-case-domains)	(update-sliding-interface-write- case)
(%import-by-filter)	(create-case-domains-of-type)	(case-has-si-face-periodics?)
(gui-write-case-data)	(default-domain-name)	(delete-sliding-interface)
(ti-write-case-data)	(%get-domain-by-name)	(recreate-sliding-interfaces)
(write-case-data)	(%get-domain-by-id)	(delete-sliding-interface-case- threads)
(read-zone-grid-data)	(get-domain)	(create-sliding-interface)
(ti-read-zone-grid-data)	(domain?)	(update-sliding-interfaces)
(gui-read-case-data)	(inquire-domain-names)	(update-si)
(ti-read-case-data)	(domain-type)	(create-sliding-interface-threads)
(read-case-data)	(domain-id)	(create-si-threads)
(gui-write-data)	(domain-name)	(build-sliding-interface-grids)
(ti-write-data)	(domains-compat)	(build-si-grids)
(write-data)	(set-sfc-values)	(build-si-compatibility)
(append-data)	mphase-zone-vars-compat list	(gui-user-fan-model)
(gui-read-data)	(get-phase-threads-with-id)	(ti-user-fan-model)
(ti-read-data)	(next-domain-id)	(register-user-fan-monitor)
(read-data)	(delete-phase-domain)	(user-fan-monitor)
(gui-write-case)	(add-phase-domain)	(update-user-fans)
(ti-write-case)	(update-domains)	(user-fan-command)
(write-case)	(domains-changed)	(user-fan-input-name)
(read-zone)	(get-domains-manage-panel)	(user-fan-output-name)
(ti-read-zone)	(get-all-domains)	(list-flow-init-defaults)
(gui-read-case)	(get-interaction-domains)	(thread-name-default)
(ti-read-case)	(get-phase-domains)	(inquire-grid-threads)
(read-case)	(get-geom-domains)	(grid-check)
(client-case-data-pattern)	(get-domains-of-type)	ti-target-mfr-menu list
(client-case-pattern)	reorder-domains n/a	threads-package list
(ti-set-file-format)	(free-domains)	(move-from-nosolve-threads)
(client-show-configuration)	(delete-domain)	(move-to-nosolve-threads)
client-run-time-release 3	(create-domain)	(delete-all-threads-of-phase)
(ok-to-discard-data?)	(free-virtual-cells)	(get-nosolve-phase-threads)
(ok-to-discard-case?)	(rehide-skewed-cells)	(get-nosolve-face-cell-threads)
(canonicalize-filename)	(ti-unhide-skewed-cells)	(get-phase-threads)
(strip-version)	(ti-hide-skewed-cells)	(get-all-threads)
(string-suffix-ci?)	(draw-shell-junction)	(get-face-threads)
(client-check-grid)	(draw-shell)	
(client-check-case)	(free-shells)	

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(get-cell-threads)	(merge-like-threads)	(read-flow-dir)
(reorder-threads)	(merge-threads)	fl-dir-z n/a
(free-threads)	(gui-separate-cell-thread)	fl-dir-y n/a
(delete-single-thread)	(ti-separate-cell-thread-by-region)	fl-dir-x n/a
(delete-thread)	(ti-separate-cell-thread-by-mark)	(ti-set-thread-name)
(%create-thread)	(gui-separate-face-thread)	(get-fluid-thread-material)
(copy-thread)	(ti-separate-face-thread-by-region)	(set-fluid-thread-material)
(create-thread)	(ti-separate-face-thread-by-angle)	(thread-type-name->object)
(create-thread-for-domain)	(ti-separate-face-thread-by-face)	(inquire-thread-names)
(gui-update-wall-thread-list)	(ti-separate-face-thread-by-mark)	(thread-name->id)
(gui-threads-copy)	(ti-slit-periodic)	(thread-id->name)
(ti-copy-bc)	(slit-periodic)	(thread-var)
(copy-thread-bc)	(ti-repair-periodic)	(thread-kind)
(set-thread-vars)	(ti-create-periodic)	(thread-type)
(get-thread-vars)	(create-periodic)	(thread-name)
(gui-threads-manage)	(gui-fuse-threads)	(thread-domain-id)
cell-type-menu list	(ti-fuse-threads)	(thread-id)
external-type-menu list	(fuse-threads)	(cell-thread?)
internal-type-menu list	(ti-orient-face-thread)	(non-periodic-boundary-thread?)
periodic-type-menu list	(orient-face-thread)	(boundary-thread?)
(pick-thread-type)	(sew-all-two-sided-walls)	(wall-thread?)
(initialize-thread-menus)	(sew-two-sided-wall)	(get-all-thread-vars)
(initialize-thread-lists)	(slit-all-two-sided-walls)	(list-threads)
(get-all-thread-types)	(slit-two-sided-wall)	(thread?)
(get-cell-types)	(ti-slit-face-thread)	(get-boundary-threads)
(get-external-types)	slit-face-thread n/a	(get-threads-of-type)
(get-internal-types)	(thread-prop)	(get-phase-thread)
(get-periodic-types)	(thread-materials)	(%get-thread-by-name)
(thread-type-ignore?)	(default-material-name)	(%get-thread-by-id)
(thread-type-index->name)	(read-thread-names)	(get-thread)
(thread-type-name->rename)	(read-cell-thread-id-list)	(cleanup-case-surfaces)
(%thread-type-name->index)	(read-boundary-thread-id-list)	(create-case-threads)
(thread-type-name->index)	(read-thread-id-list)	(mp-vars-compatible)
thread-type-list list	(read-thread-id)	(update-thread-materials)
(replace-zone)	(read-cell-thread-list)	common-phase-bc-types? #t
(ti-replace-zone)	(read-boundary-thread-list)	(sort!)
(ti-activate-cell-threads)	(read-wall-thread-list)	(sort)
(ti-deactivate-cell-threads)	(read-boundary-thread)	qsort.provided #t
(update-scheme-threads)	(read-thread-list)	(rampant-repl)
(gui-activate-cell-threads)	(read-thread)	(rampant-initialize)
(gui-deactivate-cell-threads)	(ti-set-type-thread-reference-values)	(exit-rampant)
(deactivate-cell-thread)	(ti-set-type-thread-flow-inits)	(shutdown-lam)
(ti-delete-cell-threads)	(ti-set-type-thread-vars)	(check-lam-mpi-tasks)
(gui-delete-cell-threads)	(set-thread-type!)	list-compute-nodes-process-count n/a
(activate-cell-thread)	(ti-set-thread-type)	wipe-compute-node? n/a
(delete-cell-thread)	(ti-per-pg-bc)	(isetvar)
(ti-extrude-face-thread-parametric)	(ti-per-mfr-bc)	(client-ti-set-var)
(ti-extrude-face-thread-delta)		(ti-set-var)
(extrude-thread)		(client-ti-set-window-var)
(gui-merge-threads)		(ti-set-window-var)
(ti-merge-threads)		

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(inquire-plot-info)	(rp-spe-part?)	(rpsetvar)
(update-physical-time)	(rp-spe?)	(rpgetvar)
(physical-time-steps)	(rp-sge?)	(inquire-all-versions)
(unsteady-iterate-hook)	(rp-seg?)	(gui-rampant-run)
(init-flow)	(rp-sa-des?)	(choose-version)
(init-s2s)	(rp-sa?)	(cx-gui-rsf)
(init-dtrm)	(rp-react?)	(inquire-option)
(ti-patch)	(rp-net?)	communicator #f
(ti-iterate)	(rp-lsf?)	(rampant-file-version)
(iterate)	(rp-les?)	(rampant-run)
(dynamic-mesh-suspend-	(rp-lam?)	(fluent)
surfaces)	(rp-kw?)	(cx-transform-turbo-surf)
(set-config)	(rp-ke?)	(cx-create-turbo-surf)
(dpm-cache?)	(rp-inviscid?)	(cx-set-turbo-axis)
(rp-thread?)	(rp-hvac?)	(%cx-set-average-direction)
(rp-host?)	(rf-energy?)	(cx-surface-write-values)
(rp-graphics?)	(rp-amg?)	(%surface-integrate)
(rp-double?)	(rp-dual-time?)	(%iso-zone)
(rp-3d?)	(rp-unsteady?)	(%iso-surface)
(solar?)	(rp-dpm-cache?)	(%planar-point-surface)
(sg-vfr?)	(rp-axi?)	(%point-surface)
(sg-uds?)	(rp-atm?)	(%cx-is-arc-surface)
(sg-udm?)	(rp-acoustics?)	(cx-exchange-node-values)
(sg-swirl?)	(update-cx-client-information)	(%cx-surface-fill-temp)
(sg-soot?)	(update-bcs)	(%cx-surface-get-min-max)
(sg-s2s?)	(bcs-changed)	(cx-activate-fast-iso)
(sg-rsm?)	(models-changed)	(cx-suspend-fast-iso)
(sg-rosseland?)	(inquire-version-command)	(%cx-end-fast-iso)
(sg-pull?)	(inquire-version)	(%cx-start-fast-iso)
(sg-premixed?)	(rf-cache-config)	(probe-surface-info)
(sg-pollut?)	(inquire-config)	(track-surface-on-zone)
(sg-pdf-transport?)	(client-monitor-freq-proc)	(track-surface)
(sg-pdf?)	(rp-2d?)	(%display-surface-elements)
(sg-par-premix?)	(object-parent)	(%iso-clip)
(sg-p1?)	(object?)	(%transform-surface)
(sg-mphase?)	(environment-parent)	(%cell-surface)
(sg-melt?)	(environment?)	(%zone-surface)
(sg-network?)	system-global-syntax-table list	(%surface-append!)
(sg-dynmesh?)	(syntax-table-define)	(%delete-surface)
(sg-dtrm?)	object.provided #t	(%free-surfaces)
(sg-dpm?)	(update-menubar)	(%domain-var-value-set!)
(sg-crev?)	(client-update-menubar)	(%domain-var-value)
(sg-bee-gees?)	(gui-reset-symbol-list-items)	(%rp-var-value-set!)
(sg-disco?)	(gui-menu-insert-subitem!)	(%rp-var-value)
(sg-cylindrical?)	(gui-menu-insert-item!)	(%rpgetvar)
(rp-v2f?)	(gui-not-yet-implemented)	(rp-var-clear-cache)
(rp-visc?)	clgui.provided #t	(inquire-sub-threads)
(rp-absorbing-media?)	(client-set-var)	(%separate-skewed-cells)
(rp-turb?)	(client-get-var)	(%unhide-cells)
(rp-trb-scl?)	(client-var-define)	(%hide-cells)
(rp-spe-surf?)	(domainsetvar)	(%fast-io-transfer-dumps)
(rp-spe-site?)	(domaingetvar)	(%write-network-history)

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(%network-temperature)	(%dpm-suggest-nthreads)	(%group-s2s-cells)
(%init-crevice-memory)	(%dpm-get-nthreads)	(%delete-all-s2s-groups)
(%free-crevice-memory)	(%dpm-set-nthreads)	(dtrm-rays-ok?)
(%initialize-sound-array)	(%open-isat-library)	(dtrm-set-rays-ok)
(%process-and-plot)	(%dpm-particle-summary-all)	(dtrm-globs-done?)
(%finish-fft-process)	(%dpm-particle-summary)	(dtrm-set-globs-done)
(%write-sound-pressure)	(dpm-clear-all-particles)	(%read-rays)
(%set-adjacent-cell-zone)	(dpm-inject-particles)	(%ray-trace)
(%extract-acoustics-signals)	(dpm-print-summary)	(update-storage-before-dtrm)
(%compute-acoustics-sound-pressure)	(dpm-get-summary)	(%delete-all-groups)
(%set-acoustic-read-threads)	(dpm-inquire-summary-names-sectioned)	(%group-cells)
(%initialize-acoustics-file-for-read)	(dpm-inquire-summary-names)	(display-globs)
(%set-acoustics-receivers)	(dpm-inquire-particle-functions-sectioned)	(%insert-rays)
(%read-acoustic-next-timestep)	(dpm-inquire-particle-functions)	(%display-curr-display-glob)
(%write-acoustic-data)	(dpm-inquire-particle-types)	(%change-curr-display-glob)
(%which-domain)	(dpm-list-injections)	(execute-udf-eval)
(%domain-super-domain)	(dpm-get-min-max-units)	(%repartition-remeshing-zones)
(%domain-sub-domain)	(dpm-compute-pathlines)	(%remesh-local-prism-faces)
(%set-domain-variables)	(dpm-flush-sources)	(%print-remesh-cell-marks)
(%free-phase-domain)	(%dpm-free-injections)	(%draw-remesh-cell-marks)
(%create-phase-domain)	(%dpm-delete-injection)	(%mark-remesh-cells)
(test-migrate-shell)	(%dpm-set-injection)	(%remesh-local-cells)
(%delete-shell)	(dpm-parameters-changed)	(%print-forces-moments)
(%draw-shell-junction)	(who-am-i)	(modify-lift)
(%report-shell-status)	(%contact-monitor)	(compute-lift)
(%free-shell)	(%isat-table-size)	(inquire-motion)
(%create-shell)	(%kill-isat-table)	(%find-cell-at-location)
(%create-all-shells)	(%write-isat-table)	(%prismatic-layer)
(%check-coupled-thread)	(%read-isat-table)	(%insert-cell-layer)
(hanging-or-sliding-mesh?)	(clear-pdf-particles)	(remesh-cell-region)
(host-domain-filled?)	(init-pdf-particles)	(subdivide-mesh-layer)
(solver-cpu-time)	(%free-pdf-memory)	(%refine-mesh-by-mark)
(report-connectivity)	(inquire-pdf-species)	(%contour-node-displacement)
(delete-hxg)	(%pdf-init)	(%get-max-skewness-on-zone)
(over-ride-zone)	(pdf-type)	(%get-min-max-volume-on-zone)
(get-hxc-info)	(%read-pdf)	(%check-dynamic-mesh)
(set-effectiveness-table)	(%write-pdf)	(%update-dynamic-mesh)
(set-hxc-enthalpy)	(%delete-all-sgroups)	(%update-dynamic-threads)
(get-hxc-dimension)	(%group-s-globs)	(%free-dynamic-mesh)
(%draw-macro)	(%write-surface-globs)	(%delete-dynamic-thread)
(init-hxg-model)	(%compute-solar-pos)	(%create-dynamic-thread)
(%allocate-hxg-memory)	(s2s-glob-ok?)	(%download-dynamic-threads)
(%free-all-hxgs)	(s2s-set-glob-ok)	(%init-dynamic-mesh)
(phase/total-volume)	(s2s-globs-done?)	(%create-dynamesh-node-grids)
(initialize-unsteady-statistics)	(s2s-globs-done)	(%inquire-si-parents)
(%init-disco)	(%read-sglobs-vf)	(%inquire-si-mpperiodics)
(%stop-particle-history)	(%read-sglobs)	(%list-sliding-interfaces)
(%start-particle-history)	(%sglob-info)	(%free-sliding-interfaces)
(pathline-print-summary)	(update-storage-before-s2s)	(%clear-sliding-interfaces)
(dpm-print-cache-report)	(%read-s2s-cells)	(%delete-sliding-interface)
		(%update-sliding-interfaces)
		(%update-sliding-interface)

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(%create-sliding-interface)	(%write-interp-data)	(%allocate-parallel-io-buffers)
(%clean-up-sliding-interfaces)	(%filter-data)	(%query-parallel-io-buffers)
(%sliding-mesh?)	(%update-storage-phase-walls)	(%limit-parallel-io-buffer-size)
(%non-conformal-mesh?)	(%properties-need-update)	(%free-parallel-io-buffers)
(%non-conformal-count)	(%property-methods)	(%resolve-duplicate-hanging-nodes)
(set-relaxation-method)	(%list-reactions)	(%dpm-full-locate-particles)
(%invalidate-storage)	(%free-materials)	(%dpm-node-locate-particles)
(set-residual-history-size)	(%delete-material)	(%dpm-host-locate-particles)
(%open-udrg-library)	(%create-material)	(%dpm-node-to-host-particles)
(%open-rgas-library)	(%set-material!)	(%dpm-host-to-node-particles)
(%open-udf-library)	(inquire-timers)	(%dpm-host-to-node-source)
(%user-function-list)	(print-flow-timer)	(%node-to-host-solution)
(%chip-exec)	(print-data-timer)	(%free-host-domain)
(%chip-link)	(print-case-timer)	(%fill-host-domain)
(%execute-at-end)	(end-write-data-timer)	(%dpm-fill-host-domain)
(%udf-on-demand)	(start-write-data-timer)	(%create-neighborhood)
(%chip-listing)	(end-write-case-timer)	(kill-zero-cell-compute-nodes)
(%chip-compile-file)	(start-write-case-timer)	(kill-compute-node)
(%draw-oned-cell)	(end-read-data-timer)	(%get-process-ids)
(%advance-oned-solution)	(start-read-data-timer)	(show-virtual-machine)
(%update-oned-bcs)	(end-read-case-timer)	(print-time-stamps-to-file)
(%create-oned-udfs)	(start-read-case-timer)	(print-global-timer-offsets)
(%start-oned-library)	(clear-flow-timer)	(calculate-global-timer-offset)
(%close-oned-library)	(clear-data-timer)	(mp-clear-send-recv-time-stamps)
(%open-oned-library)	(clear-case-timer)	(mp-clear-timer)
(%initialize-storage)	(print-timer-history)	(mp-wall-time)
(%sample-min-max)	(clear-timer-history)	(mp-get-tcp-chunk-size)
(%sample-bins)	(init-timer-history)	(mp-set-tcp-chunk-size)
(%delete-sample)	(clear-domain-timers)	(mp-set-comm-timer)
(%sample-list)	(%write-hosts-file)	(%mp-set-exchange-size)
(%read-sample-file)	(%read-hosts-file)	(%mp-set-socket-size)
(%get-zone-heatflux)	(%list-hosts)	(mp-debug)
(%report-zones-torque)	(%delete-host)	(mp-trace)
(%get-zone-torque)	(%delete-all-hosts)	(mp-allow-suspend)
(%initialize-nr-bcs)	(%add-host)	(mp-set-suspend)
(%display-profile-points)	(get-parallel-communicator)	(mp-set-generic-gop)
(%write-fan-profile)	(check-compute-nodes-unique?)	(%mp-set-time-out)
(%profile-list)	(%list-compute-nodes-process-count)	(prf-print-vars)
(%delete-profile)	(%list-compute-nodes)	(compute-node-count)
(%create-oriented-profile)	(update-after-migration)	(rp-host-id)
(%any-thread-has-profile?)	(%internet-dot-address)	(rp-host-set-var-cache)
(%update-mixing-plane-profile)	(prf-update-all-rpvars)	(rp-host-function)
(%create-mixing-plane-profile)	(%prf-spawn)	(probe-all-marked-cell-info)
(%update-dynamic-profiles)	(d-prf-set-var)	(probe-marked-cell-info)
(%free-profiles)	(prf-set-var)	(probe-thread-info)
(%write-cgns-export)	(prf-set-partition-mask)	(upload-thread-vars)
(%write-profile-section)	(%set-check-partition-mismatch)	(download-thread-vars)
(%read-transient-table)	(%prf-flush-message)	(%write-data)
(%read-profile-file)	(prf-exit)	(%write-surface-grid)
(%read-profile-section)	(prf-command-finished)	(%write-case)
(%inquire-interp-field-names)	(fill-any-storage-allocated)	
(%interpolate-cell-thread-data)		

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(%read-sectioned-file)	(%init-grid)	(domain-extents)
(initialize-domain-vars)	(misc-mem-stats)	(%reset-node-list)
(domain-var-default)	(mem-stats)	(%set-machine-valid)
(domain-var-units)	(display-thread-sv-data)	(%set-grid-valid)
(domain-var-value-define)	(display-thread-sv)	(%set-data-valid)
(domain-var-value-set!)	(display-memory-on-thread)	(%set-case-valid)
(domain-var-value)	(display-memory)	(data-modified?)
(domain-var-define)	(dump-memory)	(case-modified?)
(domain-var-object)	(display-infinite-storage)	(grid-modified?)
(rp-var-default)	(display-constant-storage)	(machine-valid?)
(rp-var-units)	(display-untouched-storage)	(data-valid?)
(rp-var-value-define)	(delete-untouched-storage)	(case-valid?)
(rp-var-value-set!)	(reorder-stores)	(grid-valid?)
(rp-var-value)	(display-storage)	(%save-data-id)
(rp-var-define)	(minimize-storage)	(%save-case-id)
(rp-var-object)	(display-tuples)	(update-case-id)
(inquire-release)	(garbage-collect-tuples)	(print-bandwidth)
(%residual-history)	(fluent-exec-name)	(%reorder-threads)
(thread-normal)	(fluent-arch)	(%reorder-domain)
(volume-integrals)	(%clear-domain)	(reorder-cells-by-position)
(domain-thread-integrals)	(inquire-adjacent-threads)	(%remove-orphan-bridge-faces)
(inquire-grids)	(%reset-inquire-all-adjacent-threads)	(print-partitions)
(debug-cell-and-face)		(print-domain)
(print-model-timers)	(%inquire-all-adjacent-threads)	(%merge-like-threads)
(%grid-debug)	(inquire-nosolve-face-threads)	(%free-changed-id-list)
(have-read-partitioned-case?)	(inquire-network-face-threads)	(update-thread-ids)
(%grid-check-duplicate-nodes)	(inquire-network-cell-threads)	(%scan-sectioned-file)
(%grid-check-partition-encapsulation)	(inquire-face-threads)	(%read-sectioned-zone)
(%grid-check)	(inquire-cell-threads)	(%finish-read-zone)
(%convert-data-to-absolute)	(inquire-mesh-type)	(%create-phase-threads)
(%convert-data-to-relative)	(%inquire-cell-vector-functions)	(%get-changed-ids)
(%reset-thread-element-type)	(%inquire-cell-functions-sectioned)	(%init-read-zone)
(%translate-grid)	(%inquire-cell-functions)	(%create-face-and-shadow-pair)
(%scale-grid)	(solver-statistics)	(%activate-cell-thread)
(reset-residuals)	(solver-residuals)	(%deactivate-cell-thread)
(%repair-translational-periodic)	(%repair-face-handedness)	(%delete-cell-thread)
(%repair-rotational-periodic)	(%repair-face-to-cell-threads)	(%merge-threads)
(%merge-periodic)	(%advance-particles)	(%fuse-threads)
(%split-periodic)	(%iterate-time-step)	(%extrude-face-thread)
(%inquire-equations)	(%reset-vof-solution)	(%separate-cell-thread-by-region)
(%inquire-patch-variable-names)	(%update-vof-solution)	(%separate-cell-thread-by-mark)
(patch)	(%update-physical-time)	(%separate-face-thread-by-region)
(get-thread-derived-variables)	(%iterate)	(%separate-face-thread-by-angle)
(set-thread-variables)	(flow-init)	(%separate-face-thread-by-face)
(%set-thread-type)	(fill-face-threads)	(%separate-face-thread-by-mark)
(thread-empty?)	(%get-min-max-info)	(%sew-two-sided-wall)
(thread-exists?)	(%cell-only-field?)	(%sew-all-two-sided-walls)
(update-before-data-write)	(%fill-face-thread-values)	(%slit-two-sided-wall)
(update-after-data-read)	(%fill-cell-values)	
(update-before-data-read)	(%fill-node-values)	
(%build-grid)	(thread-extents)	

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(%slit-all-two-sided-walls)	(%mark-percent-of-ncells)	(vector-function-surface)
(%slit-face-thread)	(%mark-inside-cylinder)	(velocity-vector-surface)
(%orient-face-thread)	(%mark-inside-sphere)	(%thread-grid)
(%smooth-partition)	(%mark-inside-hex)	(grid-partition-boundary)
(%reorder-partitions)	(%mark-inside-iso-range)	(draw-symmetries)
(%merge-partition-clusters)	(%mark-boundary-cells-per-thread)	(%inquire-periodic-transform)
(%copy-active-partition-to-stored)	(%mark-with-yplus-per-thread)	(draw-periodics)
(%combine-partition)	(%mark-with-refine-level)	(%partition-surface)
(%inquire-must-prettest-partition-functions)	(%mark-with-volume-change)	(%apply-slice)
(%inquire-prettest-partition-functions)	(%mark-with-volume)	(%read-cgns-data)
(%inquire-partition-picks)	(%mark-with-boundary-volume)	(update-cell-function-lists)
(%inquire-partition-functions)	(%reset-boundary-proximity)	client-monitor-solution-done #f
(%repartition-by-zone)	(%mark-with-gradients)	*remain-timestep/iter* list
(set-partition-in-marked-region)	(%not-bit-in-marked-cells)	*time/iteration* 0
(partition-count)	(%xor-bits-in-marked-cells)	(set-monitor-transient)
(partition)	(%or-bits-in-marked-cells)	(monitor-transient-solution)
(%auto-partition)	(%and-bits-in-marked-cells)	(get-monitor-frequency)
(%auto-encapsulate-si)	(%copy-pair-bits-in-marked-cells)	(show-solution-monitors)
(encapsulate-si)	(%copy-bits-in-marked-cells)	(clear-solution-monitors)
(set-case-buffer-size)	(%toggle-bit-in-marked-cells)	(cancel-solution-monitor)
(get-case-buffer-size)	(%set-bit-in-marked-cells)	(register-solution-monitor)
(print-neighborhood)	(%clear-pair-bits-in-marked-cells)	(monitor-solution-done)
(sync-solution)	(%clear-bit-in-marked-cells)	(monitor-solution-message)
(%migrate-partitions)	(%limit-marked-cells)	(monitor-solution)
(%print-migration-statistics)	(%count-marked-cells)	(client-set-version-info)
(%copy-dest-part-to-stored)	(%clear-marked-cells)	(client-exit)
(%migrate-cells-before-slide)	(%clear-pair-bits-in-all-marked-cells)	(upload-zone-vars)
(%migrate-cells-after-slide)	(%clear-bit-in-all-marked-cells)	(download-zone-vars)
(%migrate-cells)	(%clear-all-marked-cells)	(upload-cx-vars)
(%relabel-entities)	(%clear-cell-function-names)	(download-cx-vars)
(%balance-before-kill-node)	(%models-changed)	cx-windows/hardcopy-menu list
(%calculate-balanced-partitions)	(materials-require-model-change?)	cx-windows/hardcopy/driver-menu list
(%migrate-shell-back)	(%rp-config)	cx-windows/hardcopy/ps-format-menu list
(%valid-partition-id?)	(update-node-flags)	cx-windows/hardcopy/color-menu list
(generate-fpe)	(init-node-flags)	cx-windows-menu list
(cell-bins)	(%delete-turbo-topology)	cx-windows/xy-menu list
(%swap-mesh-faces)	(%define-turbo-topology)	cx-windows/video-menu list
(%smooth-mesh)	(xy-plot-turbo-avg)	cx-windows/text-menu list
(%compute-sizing-function-defaults)	(calc-turbo-avg)	cx-windows/scale-menu list
(%free-sizing-functions)	(display-turbo-avg)	cx-windows/main-menu list
(%draw-bridge-nodes)	(display-path-cone)	cx-windows/axes-menu list
(%fill-bridge-nodes)	(display-domain-labels)	(ti-set-window-pixel-size)
(%free-parents)	(display-node-flags)	(ti-set-window-aspect)
(%make-hanging-interface)	(display-marked-cells)	(ti-hardcopy-window)
(%refine-mesh-hang)	(mouse-split)	(ti-set-window)
(%adapt-mesh-hang)	(fill-cx-array)	(ti-close-window)
(%refine-mesh)	(contour-surface)	(ti-open-window)
(%adapt-mesh)		(cx-display-ppm)
		(cx-display-image)

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(cx-gui-hardcopy-options)	client-update-title-hook #f	(cx-dialog-done)
(cx-gui-hardcopy)	(cx-update-button-functions)	(cx-multiple-file-dialog)
(cx-hardcopy-suffix)	(right-button-function)	(cx-file-dialog)
(cx-hardcopy-file-filter)	(middle-button-function)	(chdir)
(cx-panelfig)	(left-button-function)	(cx-select-dialog)
(cx-graphicsfig)	(orbit-axes)	(cx-prompt-dialog)
(cx-set-small-window)	(print-marked-cell-info)	(cx-working-dialog)
(cx-preview-hardcopy)	(cx-show-probe-hooks)	(cx-yes-no-dialog)
(cx-set-window-size)	(cx-remove-probe-hook)	(cx-ok-cancel-dialog)
(cx-set-window)	(cx-install-probe-hook)	(cx-info-dialog)
(cx-close-window)	(cx-set-default-probe-hook)	(cx-warning-dialog)
(cx-display-geom-window)	(cx-mouse-probe)	(cx-error-dialog)
(cx-open-window)	(cx-mouse-orbit)	client-support-field-derivatives
(cx-use-new-window?)	(cx-mouse-dolly)	#f
(client-valid-window-owner?)	(cx-remove-text)	(client-free-host-domain)
(client-assign-window-owner)	(cx-display-label)	(client-host-domain-filled?)
cxwindows.provided #t	(cx-mouse-annotate)	(client-host?)
(ti-button-functions)	(cx-interrupt-mouse-point)	(client-max-partition-id)
(read-function)	(cx-get-mouse-point)	(client-set-cx-vars)
(ti-display-label)	(cx-highlight-selection)	(client-get-coarse-surfaces)
(open-segments)	(cx-single-send)	(client-set-coarse-surfaces)
(delete)	(cx-sendq)	client-support-coarse-surfaces?
cx-render/cell/max 0	(cx-set-io-busy-cursor)	#f
cx-render/cell/min 0	(cx-get-kill-notification-method)	(client-delete-surface)
cx-render/node/max 0	(cx-set-kill-notification-method!)	(client-create-point-surface)
cx-render/node/min 0	(cx-is-process-running?)	(client-activate-injection-
cx-render/units #f	(cx-get-current-process)	surfaces)
cx-render/domain	(cx-set-current-process)	(client-display-dpm-pathlines)
cx-render/name	(cx-command-port)	(client-compute-dpm-pathlines)
(cx-save-layout)	(cx-trace)	(client-vector-function-surface)
(cx-save-case-state)	(cx-kill-current-process)	(client-relative-vector-in-surface-
(cx-set-case-defaults)	(cx-kill)	plane)
(cx-inquire-colors)	(cx-send)	(client-relative-vector-surface)
(cx-inquire-marker-symbols-nt)	(cx-client-run)	(client-vector-in-surface-plane)
(cx-inquire-marker-symbols)	(cx-listen)	(client-vector-surface)
(cx-inquire-line-patterns)	(cx-run)	(client-contour-surface)
(cx-gui-button-functions)	*cx-timeout* 300	(client-draw-grid-zones)
(cx-add-probe-function)	*cx-trace* #f	(client-draw-grid-outline)
(cx-update-probe-function)	(cx-interrupt-client)	(client-draw-grid)
(cx-current-probe-function-	(cx-interrupt)	(client-draw-symmetries)
name)	(menu-repl)	(client-set-symmetry)
cx-button-functions list	*cx-startup-file* #f	(client-selected-symmetry-
cx-probe-functions list	*main-menu* list	planes)
cx-null-probe-name off	(cx-repl)	(client-all-symmetry-planes)
(handle-selection)	(%checkpoint)	(client-inquire-periodic)
(popup-button)	(cx-install-checkpoint-hook)	(domain-name->id)
(resize-other-windows)	(%emergency)	(client-inquire-default-domain-
(zoom-3d-window)	(cx-install-emergency-hook)	name)
(cx-add-button-to-toolbar)	(cx-exit)	(client-inquire-domain-ids-and-
(cx-refresh-toolbar)	kill-script-filename	names)
(handle-key)	/home/mirko/killfluent11300	(client-has-multiple-domains?)
cx-key-map list	(exit)	(client-support-relative-vectors?)

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(client-draw-grid-partitions?)	(client-inquire-iteration)	(gui-execute-macro)
(client-support-grid-partitions?)	(client-inquire-title)	(gui-start-macro)
(client-support-grid-levels?)	(client-inquire-release)	(cx-list-macros)
(client-add-monitor-command)	(client-inquire-version)	*cx-macros* list
(client-set-current-dataset)	(client-inquire-name)	(cx-executing-macro?)
(client-inquire-current-dataset)	*date/time-format* %b %d, %Y	(cx-execute-macro)
(client-inquire-datasets)	(cx-initialize)	(cx-stop-macro)
(client-support-multiple-data?)	(cx-dot-pathname)	(cx-start-macro)
(client-copy-node-values-to-temp)	*cx-multithread* #f	(cx-macro-open?)
(client-node-suff-temp?)	(cx-data-3d?)	(cx-gui-batch-options)
(client-fill-face-zones)	(cx-mesh-3d?)	(gui-read-journal)
(client-cell-only-field?)	(cx-3d?)	(gui-start-journal)
client-cell-only-fields list	(cortex?)	(client-exit-on-error)
(client-fill-face-zone-values)	(write-journal-file-part)	(cx-reading-journal?)
(client-fill-cell-values)	(read-journal-file-part)	(cx-read-journal)
(client-support-cell-values?)	(read-journal-file-commands)	(cx-stop-journal)
(client-fill-node-values)	(fnmatch)	(cx-start-journal)
(client-set-node-values)	(ti-read-scheme)	(cx-journal-open?)
(client-inquire-node-values)	(ti-stop-transcript)	(cx-save-recent-files)
(inquire-section-domain-list-for-cellfunctions)	(ti-start-transcript)	(cx-update-recent-files)
(inquire-domain-for-cell-vector-functions)	(ti-macro-load)	(cx-add-recent-file)
(inquire-domain-for-cell-functionssectioned)	(ti-macro-save)	(cx-enable-recent-files)
(inquire-domain-for-cell-functions)	(ti-execute-macro)	*cx-recent-files-limit* 4
(client-inquire-cell-vector-functions)	(ti-stop-macro)	(cx-write-file)
(client-inquire-cell-functions-sectioned)	(ti-start-macro)	(cx-read-file-with-suffix)
(client-inquire-cell-functions)	(ti-read-journal)	(cx-read-file)
(client-zone-name->id)	(ti-stop-journal)	(compress-filename)
(client-zone-id->name)	(ti-start-journal)	(uncompress-filename)
(client-inquire-zones-of-type)	(cx-file-type)	*uncompress* #f
(client-inquire-zone-types)	*cx-filetypes* list	*compress* #f
(client-inquire-zone-name)	(remove)	(append-file)
(client-inquire-zone-names)	(suffix)	(write-file)
(client-inquire-boundary-zones)	(basename)	(read-file-with-suffix)
(client-inquire-interior-zones)	(strip-directory)	(read-file)
(client-inquire-zones)	(directory)	(read-file-with-suffix-and-leave-port-open)
(client-inquire-bc-name)	(temp-file)	(read-file-and-leave-port-open)
(client-inquire-axis)	(cx-pause)	(%append-file)
(client-inquire-domain-extents)	(cx-set-pause-time)	(%write-file)
(client-unsteady?)	(cx-set-delay-time)	(%read-file-and-leave-port-open)
(client-solver-sm?)	*cx-pause-time* -1	(%read-file)
(client-inquire-reference-depth)	(gui-read-scheme)	(remote-file-pattern-exists?)
(client-solver-axi?)	(stop-transcript)	(find-remote-file-with-suffix)
(client-check-data)	(transcript-open?)	(find-remote-file-pattern-with-suffix)
(client-set-time-step)	(stop-journal)	(remote-file-exists?)
(client-inquire-time-step)	(journal-open?)	(client-default-basename)
	(gui-start-transcript)	(client-inquire-binary-files)
	(cx-stop-transcript)	(client-set-binary-files)
	(cx-start-transcript)	(client-support-binary-files?)
	(cx-transcript-open?)	(quote-if-needed)
	(cx-macro-load)	
	(cx-macro-save)	
	(cx-macro-define)	

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(ok-to-overwrite-remote?)	(light->on?)	(camera->projection)
(ok-to-overwrite?)	(cx-set-light)	(camera->height)
(ti-set-batch-options)	*cx-light-symbol-inactive* @	(camera->width)
(ti-exit-on-error)	*cx-light-symbol-active* (x)	(camera->up-vector)
(ti-set-answer-prompt)	*cx-light-segment* #f	(camera->target)
cx-answer-prompt? #f	*cx-max-light* 9	(camera->position)
(ti-set-overwrite-prompt)	cxlights.provided #t	(view->transform)
cx-overwrite-prompt? #t	cxdisplay.provided #t	(view->camera)
(syncdir)	cx-view-menu list	(view->name)
(dump-scheme)	cx-camera-menu list	cxview.provided #t
cx-execute-macros-quietly? #t	(cx-gui-views)	(cx-cmap-editor)
cx-exit-on-error #f	(cx-gui-camera)	(incr-cmap-name-count)
journal-file-commands list	(cx-gui-define-mirror)	(cx-set-color-ramp-range)
cxrelease.provided #t	(cx-draw-mirrors)	(new-cmap-name)
cortex-run-time-release 3	(cx-gui-define-periodic)	(cx-show-cmap-names)
cxfiles.provided #t	(cx-gui-write-views)	(cx-set-color-map)
display/set/rendering-options	(cx-write-views)	(cx-get-cmap)
list	(non-standard-views)	(cx-add-cmap)
(display/set/rendering-	(cx-read-views)	(cx-gui-cmap-editor)
options/hsm-menu)	(cx-set-camera-relative)	(ti-vector)
display/set/pathlines-menu list	(cx-compute-default-views)	(ti-display-custom-vector)
display/set/contours list	*max-stack-size* 20	(ti-add-custom-vector)
display/set/vectors list	(cx-pop-view)	(default-vector-name)
(display/set/color-ramp-menu)	(cx-push-view)	(ti-velocity-vector)
display/set/colors-menu list	(new-view-name)	(ti-profile)
(ti-set-edge-visibility)	(cx-save-view)	(ti-contour)
(pick-hsm-method)	(cx-restore-view)	(pick-cell-vector-function)
(pick-pathline-style)	(cx-get-views)	(pick-cell-function-domain)
(pick-color-ramp-from-list)	(cx-delete-view)	(pick-cell-function)
(cx-gui-annotate)	(cx-add-view)	(ti-re-scale-generic)
(scene-max-index)	(cx-default-view)	(ti-render-generic)
(scene-list-text-objs)	(extent->zmax)	(ti-zone-grid)
(text-name->segment-key)	(extent->zmin)	ti-current-vector-domain #f
(text-name->scene-index)	(extent->ymin)	ti-current-domain #f
(parse-string)	(extent->xmin)	(profile-options)
cx-pfa-fonts? #t	(extent->xmax)	(render/grid-cb)
cx-font-sizes list	(zoom-camera)	(dpm-graphics-setup)
cx-font-names list	(roll-camera)	(gui-set-grid-rendering-options!)
(cx-gui-display-options)	(pan-camera)	*sweep/domain* #f
cx-hsm-methods list	(orbit-camera)	*sweep/sub-function* #f
(cx-set-graphics-driver)	(dolly-camera)	*sweep/function* #f
(cx-show-graphics-drivers)	(camera-up-vector)	*sweep/vector-domain* #f
(describe-graphics)	(camera-target)	*sweep/vector-name* velocity
light-interp-menu list	(camera-projection)	(gui-display-sweep-surface)
(ti-toggle-headlight)	(camera-position)	(gui-display-vectors)
(ti-set-ambient-color)	(camera-field)	(gui-display-contours)
(ti-set-light)	(cx-show-camera-projection)	(gui-display-grid-colors)
cx-lights-menu list	(cx-show-camera-field)	(gui-display-grid)
(cx-gui-lights)	(cx-show-camera-up-vector)	(cx-gui-vector-options)
(id->symbol)	(cx-show-camera-target)	(gui-custom-vectors)
(light->xyz)	(cx-show-camera-position)	(cx-inquire-user-def-vector-
(light->rgb)		names)

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(customvec->z-comp)	(cx-zone-color)	(cx-create-tab)
(customvec->y-comp)	(show-thunk-list)	(cx-create-frame-tabbed)
(customvec->x-comp)	(show-title-list)	(gui-update-cell-function-
(customvec<-name)	(get-rg-title)	widget)
(customvec->name)	(get-rg-thunk)	(gui-update-cell-sub-function-
(custom-vector-function/define)	(add-thunk-to-list)	widget)
(cx-rename-vector)	(add-title-to-list)	(gui-update-cell-domain-widget)
(cx-delete-vector)	(render-generic)	(gui-update-cell-vector-function-
(cx-get-vector)	(re-render-generic)	widget)
(cx-add-vector)	custom-vector-list list	(gui-fill-cell-values-sectioned)
(gui-manage-plot)	(start-title-frame)	(gui-fill-node-values-sectioned)
(gui-update-unsectioned-cell-	(start-standard-title-frame)	(gui-fill-cell-values)
functionlists)	(cx-init-right-frame-titles)	(gui-fill-node-values)
(gui-update-domains-for-vector-	(cx-list-delete-entry)	(name->gui-function-labels)
functions)	(type-name->id)	(gui-vector-function-label-
(gui-update-domain-from-	(display-name)	>name)
section)	(cx-gui-preselect-contour-	(gui-function-labels->names)
(gui-update-domain-lists)	functions)	(gui-function-domain-list)
(gui-update-cell-function-lists)	(cx-set-viz-lists)	(gui-function-label->name)
(cx-check-toggle-buttons)	(cx-get-viz-lists)	(gui-domain-label)
(cx-cell-only-function?)	(cx-reset-viz-lists)	(gui-vector-function-label)
(cx-cell-only-field?)	(cx-write-viz-lists)	(gui-function-label)
(custom-gui-vector-function-	(cx-read-viz-lists)	(function-name->labels)
label->name)	(cx-gen-viz-name)	(vector-function-label->name)
(custom-gui-vector-function-	(cx-get-viz-all-names)	(function-label->name)
label)	(cx-get-viz-attr)	(string-downcase)
(custom-vector-function-label-	(cx-add-viz-attr)	(gui-get-selected-surface-ids)
>name)	(cx-add-to-viz-list)	(gui-get-selected-zone-ids)
(custom-velocity-vector)	cxvlist.provided #t	(gui-pick-single-list-item)
(set-display-custom-vv)	(ti-path-lines)	(gui-unpick-list-items)
(display-vector-function)	(render-path-lines)	(gui-pick-list-items)
(styled-format)	(dpm-path-lines)	(gui-update-changed-list)
(velocity-vector)	(path-lines)	*gui-name-list-width* #f
(set-display-vector-function)	(cx-reset-path-vars)	(cx-add-separator)
(set-display-vv)	(get-path-min-max-units)	(cx-add-form)
(profile)	(pick-path-cell-function)	(cx-show-symbol-list-selections)
(rgcb-profile)	(inquire-path-cell-functions)	(cx-rename-symbol-list-items)
(contour)	(num-surf)	(gui-toggle-symbol-list-
(rgcb-contour)	(max-surf-id)	selections)
(set-profile-attr)	(total-facets)	(gui-delete-symbol-list-
(set-contour-attr)	(gui-display-particle-tracks)	selections)
(set-cont-prof-attr)	(gui-display-path-lines)	(gui-add-symbol-list-selections)
(cx-update-range-vars)	*twist/max* n/a	(cx-set-symbol-list-selections)
(restore-segment-state)	*twist/min* n/a	(cx-set-symbol-list-items)
(cx-add-to-vv-list)	*cx-viz/name*	(cx-delete-symbol-list-items)
(cx-restore-render-surfaces)	cxpath.provided #t	(gui-add-selected-symbol-list-
(cx-save-render-surfaces)	cxrender.provided #t	items)
(render-grid)	cxcmapped.provided #t	(cx-add-symbol-list-items)
(cx-draw-grid-zone)	cxalias.provided #t	(cx-add-drop-down-symbol-list)
(cx-show-open-segments)	(cx-set-plot-window-id)	(cx-add-symbol-list)
(render-surface)	*max-plot-window-id* 11	(cx-create-drop-down-symbol-
(cx-reset-grid-colors)	(cx-activate-tab)	list)

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(cx-create-symbol-list)	render-specific-vars list	(cx-add-dependent)
(cx-create-pattern-selector)	(cxisetvar)	(rainbow-ramp)
(gui-add-group-zone-list)	(cx-set-state)	(fea-ramp)
(gui-get-group-list-widget)	(cx-show-state)	(red-ramp)
(gui-get-zone-list-widget)	color-list list	(green-ramp)
(gui-add-group-zone-widgets)	(cxgetvar)	(blue-ramp)
(gui-update-zone-list)	(cxsetvar)	(gray-ramp)
(gui-init-zone-list)	(cx-set-var-val-list)	(cyan-yellow-ramp)
(gui-add-zone-list)	(cx-get-var/value-list)	(blue-purple-ramp)
(client-inquire-group-names)	(cx-get-name-matching-varlist)	(purple-magenta-ramp)
client-groups list	(cx-download-vars)	(bgrb-ramp)
(cx-rename-list-items)	(cx-upload-vars)	(rgb-ramp)
(cx-panel-designer)	(cx-show-var-stack)	(bgr-ramp)
(cx-add-real-entry)	(cx-pop-vars)	(interpolate-color-ramp)
(cx-create-profile)	(cx-push-vars)	cmap.provided #t
(cx-create-draw-area)	(cx-get-active-varlist)	(summary-table-end)
(cx-create-list-tree)	(cx-get-env-varlist)	(summary-table-hline)
(cx-create-dial)	(cx-show-envstack)	(summary-table-row)
(cx-create-scale)	(cx-show-varenv-unfiltered)	(summary-table-begin)
(cx-create-drop-down-list)	(cx-varenv-status)	(summary-line)
(cx-create-list)	(cx-show-varenv)	(summary-current-level)
(cx-create-real-entry)	(cx-close-varenv)	(summary-section)
(cx-create-integer-entry)	(cx-open-varenv)	(summary-init)
(cx-create-text-entry)	(cx-var-default-value)	(summary-title)
(cx-create-toggle-button)	(cx-var-value)	(cx-tui-complex-profile-string)
(cx-create-button)	(cx-var-value-set!)	(cx-tui-complex-profile)
(cx-create-text)	(cx-var-define)	(cx-gui-complex-profile)
(cx-create-button-box)	(cx-var-object)	(cx-register-profile-method)
(cx-create-table)	(cx-var-initialize)	(cx-tui-profile-string)
(cx-create-frame)	cx-variables list	(cx-tui-profile)
(cx-add-check-buttons)	(set-unit)	(ti-menu-load-string)
(cx-add-radio-buttons)	(cx-show-units)	(ti-menu-load)
(cx-show-check-button)	(to-si-units)	(ti-menu-load-port)
(cx-set-check-button)	(to-user-units)	(ti-menu-error)
(cx-add-toggle-button)	(read-list-with-units-prompt)	(alias)
(cx-add-radio-button-box)	(read-with-units-prompt)	alias-table list
(cx-add-check-button-box)	(read-list-with-units)	(read-generic-in-range-prompt)
(cx-hide-panel)	(read-with-units)	(read-real-in-range)
(cx-create-hoops-panel)	(write-with-units)	(read-integer-in-range)
(cx-create-panel)	(cx-lookup-units)	(read-object-generic-list)
(cx-hide-item)	(ti-set-unit)	(read-object-generic)
(cx-get-item-id)	(cx-set-unit)	(read-object-id/name-list)
(cx-get-menu-id)	*cx-unit-table* list	(read-object-id/name)
(cx-update-menubar)	(cx-inquire-unit-systems)	(read-string-from-list)
(cx-delete-item)	(cx-set-unit-system)	(read-symbol-from-list)
(cx-clear-menubar)	*cx-conversion-table* list	(read-symbol-list)
(cx-add-menu)	units.provided #t	(read-symbol)
(cx-add-hitem)	cxvar.provided #t	(yes-or-no?)
(cx-add-item)	(cx-update-pending?)	(y-or-n?)
cx-sort-surface-lists #t	(cx-changed)	(read-string/symbol-list)
cx-panel-apply-close #f	(cx-show-dependents)	(read-string-list)
cxgui.provided #t	(cx-delete-dependents)	(read-boolean-list)

*This is a free-translation made by Tiago Macarios. In case of mis-translation please contact tiagom@polo.ufsc.br

(read-real-list)
(read-integer-list)
(read-filename)
(read-symbol/string)
(read-string/symbol)
(read-string)
(read-real)
(read-integer)
(ti-read-unquoted-string)
(read-generic-list-prompt)
(read-generic-prompt)
(read-real-pair-list)
(real-pair?)
(read-generic-list-pair)
(read-generic-list)
(read-generic)
(readq-generic)
(insert-menu-item!)
(ti-menu-insert-item!)
(ti-menu-item->help)
(ti-menu-item->thunk)
(ti-menu-item->name)
(ti-menu-item->test)
ti-menu-load-delay 1
menu-prompt
menu-print-always #f
(cx-check-journal)
(cx-gui-do)
(ti-text-processing?)
(menu-get)
(menu-do-1)
(menu-do)
(ti-info)
(ti-menu-print)
(ti-read-default?)
(ti-input-pending?)
(ti-strip-blanks)
(ti-flush-input)
_ 0
(string->valid-symbol)
(flush-white-space)
(flush-char-set)
(read-delimited-string)
char-set:newline list
char-set:whitespace list
(char-set)
journal-file-count 0
iface.provided #t
cx.provided #t
cx-disclaimer WARNING This
is a prototype version that has
not yet

been tested and validated.
Fluent Inc.
makes no commitment to
resolve defects
reported against this prototype
version.
However, your feedback will
help us improve
the overall quality of the
product.
client.provided #t