How to read the documentation of the commands

Summarized:

This note is a guide of reading of the U4 booklets and U7 of the Instruction manual.

She explains in particular the meaning of the méta-characters and the typographical conventions used for the description of the syntax of the commands.

All examples given here are set as illustration and do not replace the complete description of the commands appearing in the booklets U4 and U7.

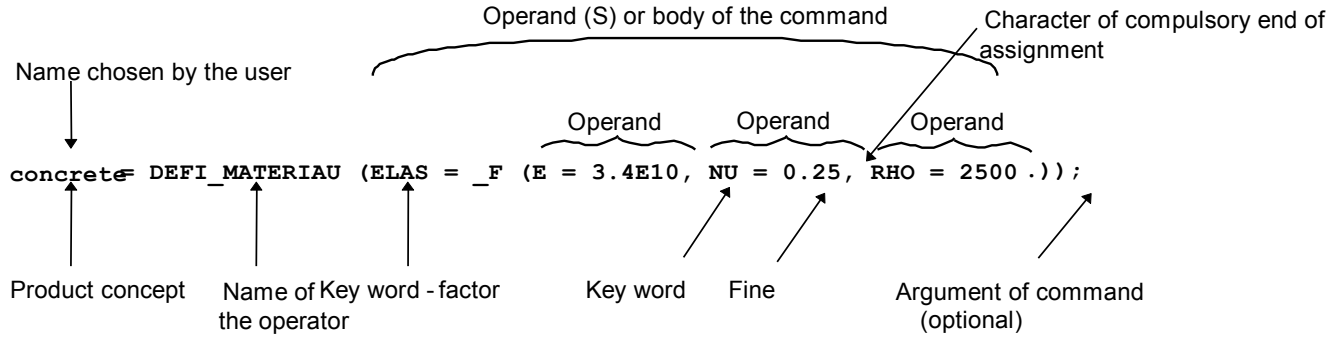
Contents

1 Recalls on syntax of the commands of Code_Aster

the process control language and its supervisor are completely described in the document [U1.03.01]. One recalls here some notions on syntax of the commands of Code_Aster .

In *Code_Aster*, one understands by the generic term of commands at the same time **the operators**, **the procedures** and the macro-commands of the process control language. An operator provides **a product concept** typified (by the operator) and named by the user. A procedure does not generate a product concept, it achieves **actions** such as printings or resource allocations.

In the example below, one recalls the vocabulary which is used in the description of the commands.



Terminology *Aster*

an operand is thus the whole consisted a key word and its argument. However, in the documentation of the commands, one often indicates the operands of an operator or a procedure by the name of their key word. For example: RHO, simple key word, or ELAS, key word factor.

The term of **product concept** is generic for all the operators, it is result work of the operator.

Here in example DEFI_MATERIAU, there was creation of data structure of the type MATER (material), named concrete by the user. It gathers the denominations and the (key word E , NU , RHO) values (arguments 3.4E10, 0.25,2500 .) of the mechanical elastic characteristics (factor key word ELAS) of the material.

The term of concept **of the type result** applies to the outputs of the operators of computation, i.e. physical fields of variables (displacements, temperatures, stresses, forces, modes, etc…) on the nodes or meshes at various times or for various frequencies.

The result concept comprises in general **under types.**

2 Standard plane of the documents of use of the commands

Each document of presentation of a command comprises the following chapters:

- Goal,
- Syntax,
- Operands,
- Examples (possibly).

This presentation makes it possible to the user to find in only one document all knowledge necessary to the implementation of a command.

3 Paragraph Drank

One states the functionality filled by the command (actions carried out). One also specifies the types of the expected concepts as starter and the product concept, as well as characteristics of the command.

This paragraph is also displayed by the search engines; it thus contains only text without equations or formula.

Example: Operator STAT_NON_LINE [U4.51.03]

Drank :

In nonlinear to calculate the quasi-static mechanical evolution of a structure.

Nonthe linearity is related either to the behavior of the material (for example plastic), or with the geometry (for example in large displacements). To have details on the method of resolution employed, one will refer to documentation of reference [R5.03.01].

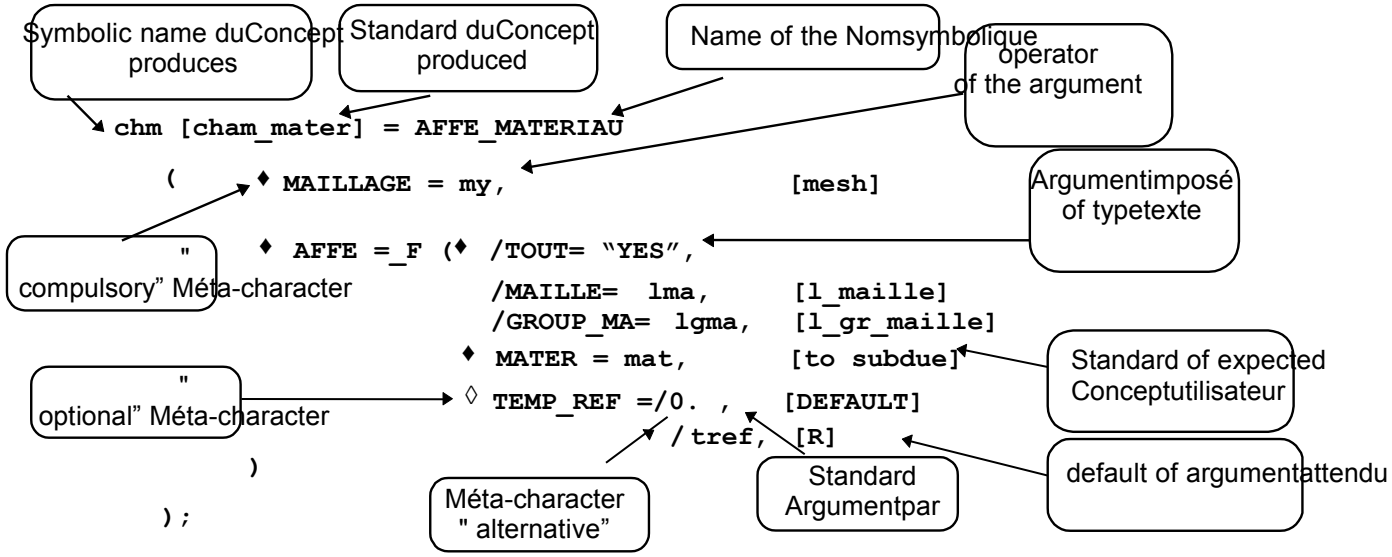
The evolution can be studied in several successive works (concept reentrant), either in poursuite (the calculated last moment is the initial time of following computation), or recovery some on the basis of one former time.

If the time necessary to carry out computation is not sufficient, the program stops, but the already calculated results are saved if a data base were defined in the profile of study of the user. Product a data structure of the evol_noli type.

4 Paragraph Syntax

One gives, in this paragraph, all the operands of the command. One specifies, for each operand, using méta-characters and of indentations suitable for the typographical presentation of the commands (cf example of operator AFFE_MATERIAU):

- the name of the operator,
- the name of the key words,
- symbolic names user of the product concept and the arguments of the key words,
- compulsory or optional character of the operands (statute),
- the alternatives in the choices of the operands,
- the standards of the arguments expected by the key words,
- the values by default taken by the arguments in the case of optional operands,
- the standard of the product concept, when it is about an operator.



Presentation of the syntax (partial) of operator AFFE_MATERIAU

4.1 Méta-characters of statute of operands (♦ ◊ / |)

Four méta-characters are used to indicate the statute of the operands. It is necessary to understand here by statute of the operands their compulsory or optional declaration and the nature of the alternatives in the choices of the operands.

These méta-characters are not part of the process control language. They have only one function of documentary presentation and do not have to thus be used for the drafting of the command file.

4.1.1 Compulsory or optional operands

They are located by the presence at the top of a black or white rhombus.

♦ black rhombus: it is compulsory to declare in the command the operands which follow this sign.
◊ white rhombus: the declaration of the operands which follow this sign is optional. In the event of absence of the operand, the command will affect possibly one or of the values by default.

Example: operator DEFI_LIST_ENTI (definition of a list of strictly increasing integers whose values are regularly spaced)

```
Li =DEFI_LISTE_ENTI
                (    ♦ debut = deb. ,
                   ◊INTERVALLE    = _F  (   ♦JUSQU_A=if            ,
                                            ♦PAS=ipas              ,
                                        )
                 )
```

- It is compulsory to declare the operand identified by the key word debut and to provide deb. which is the first integer of the list to be built.
- It is not compulsory to declare by the key word the operand identified factor INTERVALLE. In this case the list of integers will be summarized with only one integer of value deb. (this is specified in the description of the operands).
- If operand INTERVALLE is declared, then it is compulsory to declare the operand JUSQU_A which specifies the whole end yew of the interval to be cut out with a constant step and the operand NOT which indicates the step ipas interval division.

4.1.2 Alternatives in the choice of the operands

They are located by the presence at the top of each choice of the alternative:

- of one / (slash): exclusive alternative, only one choice among those proposed,
- of one | (pipe, semi colonist): nonexclusive alternative, one or more choice among those proposed.

Example of exclusive alternative: operator AFFE_MODELE (assignment of the type of finite elements on whole or part of a mesh).

```
Mo   = AFFE_MODELE (
       ♦MAILLAGE      = my
       ♦AFFE      = _F   (                    ♦/TOUT=' OUI',
                                 /MAILLE           =mail    ,
[l_maille]
                                 /   NOEUD=noeu             ,
[l_noeud]
                                 /GROUP_MA     =g_mail      ,
[l_gr_maille]
                                 /GROUP_NO     =g_noeu      ,
[l_gr_noeud]
                .............
                          )
                             )   ;
```

In operand AFFE (compulsory) it should be indicated where will be affected, on the mesh, the type of finite element specified in operands PHENOMENE and MODELISATION of the same command:

- either on all the mesh (TOUT),
- or on some meshes (MESH),
- or on some nodes (NOEUD),
- or on certain mesh groups (GROUP_MA),
- or on certain nodes groups (GROUP_NO).

Example of nonexclusive alternative:

operator AFFE_CHAR_MECA operand DDL_IMPO (assignment of displacements imposed on degrees of freedom).

```
DDL_IMPO = _F  ( ♦/TOUT         =       ' OUI',
                      /NOEUD      =   lno,       [l_noeud]
                      /GROUP_NO    =  lgno,      [l_gr_noeud]
                     /MAILLE     =lma        ,       [l_maille]
                     /GROUP_MA  =lgma     ,       [l_gr_maille]
                  ♦  |   DX  =UX              ,       [R]
                     |   DY  =UY              ,       [R]
                     |   DZ  =UZ              ,       [R]
                     |   DRX =          □ X   ,    [R]
                     |   DRY =          □ there  ,    [R]
                     |   DRZ =          □ Z   ,    [R]
                     |   GRX =G            ,       [R]
                     |   PRES=p            ,       [R]
                     |   PHI =          □ ,       [R]
                     |   TEMP=T            ,       [R]
                     |   PRE1=pr1              ,       [R]
                     |   PRE2=pr2              ,       [R]
                )
```

In this operator, it is necessary to specify obligatorily:

- the scope of application on mesh: everywhere (TOUT), on some nodes (NOEUD) or certain nodes groups (GROUP_NO),
- on which degrees of freedom with which specified values by the user.

Méta-character | indicate that the user can impose a value of displacement on **one** (the symbol ♦ indicates that one needs at least one of them) or **more** of the degrees of freedom (DX, DY, DZ, DRX, DRY, DRZ, GRX, NEAR, PHI, TEMP, PRE1, PRE2) of the beforehand indicated nodes.

4.1.3 Combinations of the méta-characters of choice of the operands

These méta-characters can be combined to illustrate the multiplicity of the choices in some commands.

Example: order DEFI_MATERIAU (definition of a material by its properties of behavior)

For a study of thermomechanics, one needs to define a material having **at the same time** mechanical characteristics (ELAS) and thermals (THER) from where use of the pipe: |

But in each choice, one is obliged to choose if the properties of the material are dependant (_FO) or not on the temperature from where use of the slash: /; cf below:

```
my =   DEFI_MATERIAU     (      |  / ELAS = _F   (  ♦   E =       yg,
                                                    ♦  NU =nu     ,
                                                    ◊  RHO =rho      ,
                                                    ◊  ALPHA =dil    ,
                                                 )
       ......
                                  / ELAS_FO =_F (  ♦   E =      f1,
                                                    ♦  NU =f2     ,
                                                    ◊  RHO =f3       ,
                                                    ◊  ALPHA =f4 ,
                                                )

                                  |  / THER = _F   ( ♦    RHO_CP =      CP,
                                                    ♦    LAMBDA =la   ,
                                                )
       ......
                                  / THER_FO =_F (    ♦    RHO_CP =g1   ,
                                                   ♦LAMBDA    =g2   ,
                                                )
       ......
                          );
```

4.2 Méta-characters of the type of concept or argument

Like the méta-characters of statutes of operands, the hooks [] and star * are not part of the process control language. They have only one function of documentary presentation.

4.2.1 Types of concepts or arguments []

They frame the type of the product concepts as well as the type of the arguments.

Example: command AFFE_MODELE (Assignment of the finite elements on meshes of a mesh)

```
Mo   [model] = AFFE_MODELE
                    (   ♦MAILLAGE  =ma   ,                        [mesh]
                        ♦AFFE   =_F    ( ♦/TOUT=           "OUI',
                                           /MAILLE        = mail,   [l_maille]
.....................
        )
```

In the example above, one thus specifies that the product concept by AFFE_MODELE is of model type and that the expected concept as argument of the key word MESH must be of l_maille type (i.e list of mesh). Type

4.2.2 of the product concept [*] This

méta-character indicates that the type of the product concept, or under type of the product concept of the type result, depends on the types of the arguments of certain operands. In this case the various possibilities are registered after syntax of the command. Example:

order CREA_CHAMP In

this example, CH 2 will be a field at nodes, a card or a field by element according to the value of TYPE_CHAM . CH

```
2 [*]   = CREA_CHAMP (
     ♦   TYPE_CHAM    = "NOEU_xxxx", [kN                  ]/
                         "CART_xxxx",/
                         "ELGA_xxxx",/
                         "ELNO_xxxx",/
                         "ELEM_xxxx",)
     SiTYPE

    _CHAM='              NOEU_TEMP_R' then          [*]  =CHAM_NO_TEMP_R
 "NOEU
                          _DEPL_R'                       CHAM_NO_DEPL_R…
                        "CART
                        _TEMP_R'                       CARTE_TEMP_R "CART
                        _DEPL_R'                       CARTE_DEPL_R…
                        Comments
```

4.3 For

some commands complex such as AFFE_CARA_ELEM or DEFI _MATERIAU for example , the character of comment is employed to comment on the alternatives of the operands. It has the same meaning that in the process control language and is interpreted like such by the supervisor. Example

for AFFE_CARA_ELEM : POUTRE

```
=_F (♦/MESH    =lma      , [l_maille                          ]/
              GROUP_MA =lgma       , [l_gr_maille                          ] ♦
          /SECTION       = "GENERALE",/#
                    constant section ◊
                    CARA=  | " A " |
                              "IY " | "IZ " lists
                           AY " | "AZ " possible
                       EY " | "EZ " constant
                       JX " |
                              "RY " | "RZ" | "RT",/#
                    variable section ◊
                    CARA=  | " A1 " |  "A2 " |
                              "IY1 " | "IY2 " | "IZ1 " | "IZ2 " lists
                           AY1 " | "AY2 " | "AZ1 " | Possible " AZ2"
                       EY1 " | "EY2 " | "EZ1 " | "EZ2 " variable
                           JX1 " | "JX2 " |
                              "RY1 " | "RY2 " | "RZ1 " | "RZ2 " | "RT1 " |
```

choices| "
for one| "
section| "

choices| "
for one| "
section| "

"RT2 ", ...
)

Standard

4.4 of the arguments expected by the key words

the key words of the operands expect arguments which correspond, in general, with four classes:

- values, one then specifies by a symbolic name the accepted data-processing type (real, whole, character string, etc…),
- imposed texts, then the texts ("OUI', "HY1 ") are indicated between quotes, of
- the names of topological entities simple (name of node, of meshes, or lists of names), declared in mesh file, or of the names of nodes groups or meshes, or lists of names of nodes groups or meshes,
- the names and the lists of names of product concepts by the operators.

The table below gathers all the principal types of the arguments expected by the keywords: 1)

[R] real	3.	[L
_R] list	of realities (1	. , 3. , 7.) [I
] whole	7	[L
_I] list	of integers (9	, 6,1,9) [C
] complex	IH	1.1,7.8 or MP 10. , 1.57 [L
_C] list	of complexes (IH	1.1,7.), (IH 4.7,9.) [TXM
] unconstrained	text (name of TITER…) "	**my title" [kN**
] text	lower or equal to N characters "	INST" [L
_Kn] list	of texts lower or equal to N characters ("	SIXX", "SIYY", "SIXY") [node
] name	of node N23	[L
_noeud] lists	names of nodes (N	23, N24, N25) [gr.

_noeud] name	of nodes group NBORD	6 [L
_gr_noeud] list	of names of nodes groups (NBORD	, NBASE, NBORD) [mesh
] name	of mesh M34	[L
_maille] lists	name of mesh (M	34, M35) [gr.
_maille] name	of mesh group MPIQUAGE	[L
_gr_maille]	names of mesh groups (MSOM	, MDROI, MGA) [standard
lists		
_concept]	of concept (or field) produced	[L
standard	beforehand with generally automatic checking of the type monresu	
_type_concept]	of the type of concept user (resu	1, resu2) Standard
list		

4.5 of the product concepts in Aster *One*

uses the méta-character of choice of exclusive alternative/to mean the plurality of concept expected behind a key word. Example:

operator ASSE _MATRICE (assembly of the elementary matrixes contained in a list of concepts of the type matr _elem_*.) my

```
[  matr_asse_*] = ASSE_MATRICE (
                ♦  MATR_ELEM    =lmel    ,/[          l_matr_elem_DEPL_R]/[
                                                      l_matr_elem_DEPL_C]/[
                                                      l_matr_elem_TEMP_R]/[
                                                      l_matr_elem_TEMP_C]/[
                                                      l_matr_elem_PRES_R]/[
                                                      l_matr_elem_PRES_C]…
);
                if
```

matr _ELEM [MATR _elem_DEPL_R] then [*] (DEPL_R [matr
_elem_DEPL_C] □ DEPL _C [matr
_elem_TEMP_R] □ TEMP_R [matr
_elem_TEMP_C] □ TEMP _C [matr
_elem_PRES_R] □ PRES_R [matr
_elem_PRES_C] □ NEAR _C In

L" example above the concept expected in argument of MATR_ELEM can be various types and type of the last concept in argument by the user will by the operator depend (according to stated rules Ci - above) typing on product concept ASSE _MATRICE. Paragraph

5 Operands One

describes, for each operand the meaning of the operand for this command, the nature and the type of the arguments expected by the key words, and the restrictions and difficulties of employment. For example

, in the documentation of operator AFFE_MATERIAU , for the operand AFFE , operand intended to specify on which (S) entity (S) topological (S) of the mesh of name my will be affected the material of name mat produces by the operator DEFI _MATERIAU, one will read: ♦

AFFE Factor key word

which makes it possible to affect various materials on “pieces” of the mesh. /TOUT

= “YES”, This

key word makes it possible to affect on all meshes mesh. /GROUP_MA

= lgma, This

key word makes it possible to affect on a list of mesh groups of the mesh. /MAILLE

=lma , This

key word makes it possible to affect on a list of meshes mesh. A

each mesh group, (key word GROUP_MA) or each list of meshes (key word NETS), or with all the mesh (key word TOUT) is affected a material mat , which is a product concept by one of operators DEFI _MATERIAU [U 4.43.01] or DEFI _COMPOSITE [U 4.42.03]. If

a mesh appears explicitly (or implicitly) in several occurrences of factor key word the AFFE , the rule of overload is observed: it is the last assignment which precedes [U2.01.08]. Phases

6 of checking/execution

the paragraph Syntax of the documentation of use is the exact reflection of the catalog of the command. This catalog is a file which understands, written in the language of the supervisor, all the rules on the keywords: presence, exclusion, implication, contained…

editor EFICAS exploit this catalog of command and allow so the user, with final the made up file is valid, to obtain a correct command set. With

the execution of the study, the supervisor of Code_Aster *reproduces* the same task of syntactic checking: either overall for all the file, or while alternating with the execution, orders by command. Moreover

, during the execution itself of the commands (entered part FORTRAN of the source code), of the additional checks can be made. They are stresses impossible to manage on the level of the process control language (equality of cardinals of different lists…). Print

7 and indentations For

the legibility of the documents concerning to the commands, all that refers to syntax is printed in police Courier 10 points . One differentiates different the element types functional ones (product concept, key word, factor key word, argument) by the use from capital letters and tiny. In capital letters

: names
- of the operators, of the procedures names
- of the key words and the key words factors, imposed
- arguments of standard text (those are between “quotes” as in the syntax of the commands). In small letters

: names
- of the product concepts, symbolic names
- of the arguments, types
- of the product concepts and the arguments. Into

mixed tiny - capital letter when the product concept admits under type. This one appears in capital letters as well as type FORTRAN of the quantity of under type. One

reinforces the legibility of syntax by the use of indentations. They are used with the location of the blocks of operands and the release of a group as operands under one factor key word. Are also used they to lay out the brackets of the same block under the same balance. Example:

my

```
[  matr_asse_*] = ASSE_MATRICE (

     ♦  MATR_ELEM     =lmel     ,/[                    l_matr_elem_DEPL_R]/[
                                                       l_matr_elem_DEPL_C]/[
                                                       l_matr_elem_TEMP_R]/[
                                                       l_matr_elem_PRES_C] ♦

        NUMERICAL     _DDL =nu   ,      [numérique_ddl
   ] ◊

        CHAR_CINE  =lcha     ,/[                    l_char_cine_meca]/[
                                                       l_char_cine_ther]/[
                                                       l_char_cine_acou] ◊

        INFO    = /1               , [DEFAULT                ] /2
                                   ,   );

   if


matr _ELEM [     matr_elem_DEPL_R] then         [*]       DEPL_R
     [matr
                 _elem_DEPL_C] DEPL_C                                    [matr
                 _elem_TEMP_R] TEMP_R                                    [matr
                 _elem_PRES_C] PRES_C
```