How to read Code_Aster Documents

Introduction:

This document is a guide for reading all the documents starting with U4 and U7.

In this document, you will find definition and meaning of the meta-characters and the typographic convention which are used for the description of the command and their syntax.

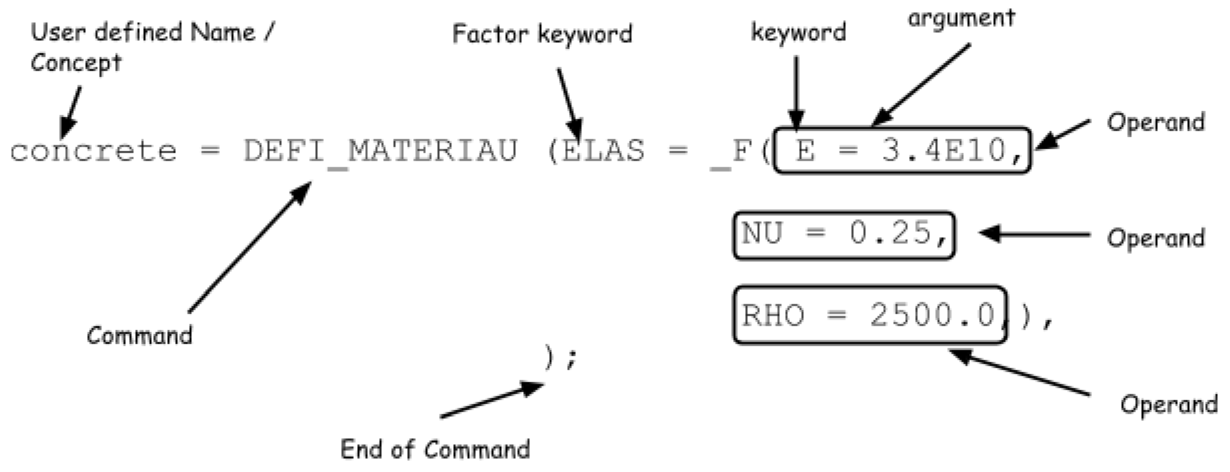
Any example given here is only for the purpose of illustration. Please refer to relevant document for accurate implementation and use.

Table of Contents:

Command Syntax used in Code_Aster

Document U1.03.01 describes the process control language and its supervisor. Some notation on syntax of commands of Code_Aster are shown for illustration purpose.

Commands are used in .comm files to either inform Code_Aster or to get information out of Code_Aster.



```
concrete = DEFI_MATERIAU (ELAS = _F( E = 3.4E10,
                                     NU = 0.25,
                                     RHO = 2500.0),
                         );
```

Operand consist of a keyword and associated argument. There are two types of keywords, simple and Factored. RHO in the above example is a Simple keyword whereas ELAS is a Factored keyword.

It is not necessary that all commands produce a Concept. There are commands viz. IMPR_RESU which do not produce any Concept.

Standard layout of the document showing use of the command

Each document showing use of the command has following sections

- Purpose
- Syntax
- Operands
- Example (optional)

This makes it possible for the user to find all relevant knowledge necessary for that command in a single document.

Section for Purpose

This section describes the functionality of the command. Specifications for the types of concepts expected for the input and the product concept, as well as characteristics of the commands will also be described in this section.

Example: Operator `STAT_NON_LINE` [U4.51.03]

Compute quasi-static mechanical evolution of a structure into nonlinear.

Note that linearity is related either to the behavior of the material (e.g. plastic), or with the geometry (e.g. large displacements). To have details on the method of resolution employed, refer to reference document [R5.03.01]

The evolution can be studied in several successive works (concept reentrant), either in continuation (the computed last moment is the initial time of following computation), or recover some on the basis of a former time.

If the time necessary to carry out computation is not sufficient, the program stops but the already computed result is saved if a data base were defined in the profile of study of the user.

Section for Syntax

In this section all operands related to the command are given. Indentation and symbols are used for typographical representation of the command for each operand.

- the name of the operator
- the name of the key word
- Product concept's symbolic names used and the arguments of the key words
- Compulsory or optional character of the operands
- Choice of alternative operands
- the standards of the arguments expected by the key words
- default values taken by the arguments in the case of optional operands
- the standard of the product concept, when it is about an operator.

Example: `AFFE_MATERIAU`

```
material [cham_mater] = AFFE_MATERIAU
     (    ◆ MAILLAGE = mesh,                         [mesh]
          ◆ AFFE = _F ( ◆ / TOUT = 'OUI',
                          / MAILLE = lma,            [l_maille]
                          / GROUP_MA = lgma,         [l_gr_maille]
                        ◆   MATER = CS,              [subdue]
                        ◇   TEMP_REF = / 0.,         [DEFAULT]
                                       / tref,       [R]
                       ),
     );
```

Presentation of the syntax of operator AFFE_MATERIAU

Symbols of status used before operands (◆ ◇ / |)

Four symbols are used before the operand to indicate the status of the operand. It is important to understand the status of the operand, whether it is compulsory or optional in nature and whether it is an alternative.

These symbols are not a part of the control language. They are shown for easy understanding of the document only and should not be used in the final command file.

Compulsory or optional operands (◆ ◇)

Operands are identified as compulsory or optional by the presence of a black or white rhombus before them.

◆ A black rhombus means that the operand that follows is compulsory
◇ A white rhombus means that the operand that follows is optional

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

```
Li = DEFI_LIST_ENTI
     (    ◆ DEBUT = deb,
          ◇ INTERVALLE = _F ( ◆ JUSQU_A = if,
                              ◆ PAS = ipas,
                             ),
     );
```

In the above example

- `DEBUT` is a compulsory operand to be declared and so is `deb` which is the first integer of the list to be built.
- `INTERVALLE` is not a compulsory operand (In this case, the list of integers will be summarised with only one integer of value `deb)`.
- If operand `INTERVALLE` is declared, then it is compulsory to declare the operand `JUSQU_A` which specifies the integer end and `PAS` which declares the internal division

Alternative Operands

Operands are identified as Alternatives by the presence of a forward slash or pipe symbol before them.

/ A forward slash denotes exclusive alternative which means that only that operand is sufficient
| A Pipe symbol denotes non-exclusive alternative which means that you can use all or some of the choices presented.

Example: 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 = ⌀Y,                              [R]
              | DRZ = ⌀Z,                              [R]
              | GRX = G,                               [R]
              | PRESS = p,                             [R]
              | PHI = ⌀,                               [R]
              | TEMP = T,                              [R]
              | PRE1 = pr1,                            [R]
              | PRE2 = pr2,                            [R]
      );
```

In the above operand DDL_IMPO it is necessary to specify that the scope of the DDL_IMPO is

- either for the whole mesh `TOUT = 'OUI'`
- or on certain Nodes `NOEUD = lno`
- or on certain Node Group `GROUP_NO = lgno`
- or on certain part of Mesh `MAILLE = lma`
- or on certain Mesh Group `GROUP_MA = lgma`

Whereas Symbol | represents that user can impose a value of displacement on one (atleast one value is required as specified by the symbol ◆) or more degrees of freedom (`DX, DY, DZ, DRX, DRY, DRZ, GRX, PRES, PHI, TEMP, PRE1, PRE2`)

Combination of the symbols of the operands

These symbols can be combined to illustrate their use in a complex manner for a command.

Example: `DEFI_MATERIAU` (Definition of the material by its properties)

For a study of thermomechanics, the user needs to define mechanical (`ELAS`) as well as thermal (`THER`) properties at the same time which is shown by the use of Pipe symbol |.

But in each choice, the user is bound to choose if the properties of the material are dependent (`_FO`) or not on the temperature which is shown by the use of Forward slash symbol /.

```
steel = DEFI_MATERIAU
      (     | / ELAS = _F ( ◆ E = e,
                            ◆ NU = nu,
                            ◇ RHO = rho,
                            ◇ ALPHA = alpha,
                          ),
              / 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,
                          ),
      );
```

Symbols for concept or argument

Similar to the symbols discussed above for the operands, the hooks [] and the star * symbol do not form part of the process control language. They have only one function of documentary presentation which is discussed in the following section.

Types of concept or arguments []

[] shows the type of the product concept or the type of the arguments after which it is specified.

Example: AFFE_MODELE (Connection of the finite elements on the meshes)

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

In the above example, it is specified that the type of the product concept AFFE_MODELE is model and that the expected concept as arguments of the key word MESH must be of the type l_maille (list of mesh).

Type of the product concept [*]

This symbol indicates that the type of the product concept, or the result, depends on the types of the arguments of certain operands. If this is the case, various possibilities are registered after the syntax of the command.

Example: CREA_CHAMP

In this example ch2 is the field with the nodes or a card or a field by elements according to the value of TYPE_CHAMP

```
ch2 [*] = CREA_CHAMP
     (     ◆ TYPE_CHAM = / 'NOEU_xxxx',                    [kN]
                         / 'CART_xxxx',
                         / 'ELGA_xxxx',
                         / 'ELNO_xxxx',
                         / 'ELEM_xxxx',
     );

          if TYPE_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

For certain complex commands, such as AFFE_CARA_ELEM or DEFI_MATERIAU, the # symbol is employed to comment on the alternatives of the operands. It has the same meaning in the process control language (which is a python file) and is interpreted like a comment by the supervisor.

Example: AFFE_CARA_ELEM

```
POUTRE = _F ( ◆ / MAILLE = lma,                         [l_maille]
                / GROUP_MA = lgma,                      [l_gr_maille]
             ◆ / SECTION = 'GENERALE',
                          / # constant section
                          ◇ CARA = | 'A',
                                   | 'IY', | 'IZ',
                                   | 'AY', | 'AZ',
                                   | 'EY', | 'EZ',
                                   | 'JX',
                                   | 'RY', | 'RZ', | 'RT',
                          / # variable section
                          ◇ CARA = | 'A1', | 'A2',
                                   | 'IY1',| 'IY2',| 'IZ1',| 'IZ2',
                                   | 'AY1',| 'AY2',| 'AZ1',| 'AZ2',
                                   | 'EY1',| 'EY2',| 'EZ1',| 'EZ2',
                                   | 'JX1',| 'JX2',
                                   | 'RY1',| 'RY2',| 'RZ1',| 'RZ2',
                                   | 'RT1',| 'RT2',
             );
```

Argument types expected by the key words

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

- Values - are specified by symbolic name and accepted data type (real, whole, character string etc)
- Imposed text - are specified between quotes (`'OUI', 'HY1'`)
- Names of the simple topological entities as declared in the mesh file, or of the names of node groups or meshes, or list of names of node groups or meshes
- The names and the list of names of product concept by the operators.

The table below shows all the main types of the arguments expected by the key words.

[R]	real number	3.
[l_R]	list of real numbers	(1., 3., 7.,)
[I]	Integer	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 Complex	(IH 1.1,7.8), (IH 4.7,9.)
[TXM]	Unconstrained text (name of TITLE)	'my_title'
[KN]	text lower or equal to N characters	'INST'
[l_KN]	list of text lower or equal to N characters	('SIXX', 'SIYY', 'SIZZ',)
[node]	Name of node	N23
[l_noeud]	List of names of nodes	(N23, N24, N36,)
[gr_noeud]	Name of node groups	NBORD6
[l_gr_noeud]	List of names of node groups	(NBORD, NBASE, NFILL,)
[mesh]	Name of mesh	M32
[l_maille]	List of names of mesh	(M32, M43,)
[gr_maille]	Name of mesh group	MPIQUAGE

`[l_gr_maille]`	List of names of mesh group	`(MSOM, MDROI, MGA,)`
`[type_concept]`	type of concept (or field) produced beforehand with generally automatic checking of the type	`resu1`
`[l_type_concept]`	List of type of concepts	`(resu1, resu2,)`

Types of product concepts in Aster

Symbol of exclusive alternative (forward slash /) is used to mean multiplicity of concept expected behind a keyword.

Example: `ASSE_MATRICE` (assembly of the elementary matrices contained in a list of concepts of the `MATR_ELEM_*` type)

```
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]          = PRES_C
```

In the above example the concept expected in argument of `MATR_ELEM` can be various types and type of the last concept in argument by the user will depend (according to stated rules above) on the product concept by the operator `ASSE_MATRICE`.

Section for Operands

In this section, for each operand, the meaning of the operand is described, the nature and the type of the arguments expected by the keywords are also described and any restrictions or difficulties faced when used.

Example: In the documentation of the operator AFFE_MATERIAU, the operand AFFE, would be described as

◆ `AFFE`

Keyword factor which makes it possible to affect various materials on sections of the mesh.

```
/TOUT = 'OUI',
```
This will affect the whole mesh
```
/GROUP_MA = lgma,
```
This will affect on the list of mesh group
```
/MAILLE = lma,
```
This will affect on the list of meshes.

Each mesh group, (key word `GROUP_MA`) or each list of meshes (keyword `MAILLE`), or with the entire mesh (keyword `TOUT`) affects a material `chechmate`, which is a product concept by one of operators `DEFI_MATERIAU` [U4.43.01] or `DEFI_COMPOSITE` [U4.42.03].

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

Phases of checking / of execution

Paragraph syntax of the documentation of us is the exact reflection of the catalogue of the command. This catalogue is a file which understands, written in the language of the supervisor, all the rules on the keywords: presence, exclusion, implication, contained …

EFICAS exploits this catalogue of command and allows the user 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, while alternating with the execution, or by the order of the commands.

Moreover, during the execution of the commands (FORTRAN source code), part of the additional checks can be made.

For Printing

Commands are printed in `Courier New` font with indentations for ease of readability. Upper cases and lower cases are used to differentiate various types of functional elements (product concept, keyword, keyword factor, argument).

What is in CAPITAL LETTERS

- Names of the operators, Names of the procedures
- Names of the keywords and the keyword factors,
- Imposed arguments of standard text (those are between quotes as in the syntax of the commands).

What is in small letters
Names of the product concepts,
Symbolic names of the arguments,
types of the product concepts and the arguments.

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_TEMP_C]
                                     / [l_matr_elem_PRES_R]
                                     / [l_matr_elem_PRES_C]

           ◆ NUME_DDL = nu,            [nume_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
```