

## Rules concerning the writing of the catalogues

---

### Summary:

We give in this document, some rules (or advices) which must comply with the developer when he adds or modifies a catalogue of order or a catalogue of finite element.

## Contents

---

<a href="#">1 Catalogues of orders.....</a>	<a href="#">3</a>
<a href="#">2 Catalogues of elements.....</a>	<a href="#">3</a>
<a href="#">2.1 Catalogue phenomenons_modelisations.py.....</a>	<a href="#">3</a>
<a href="#">2.2 Catalogue physical_quantities.py.....</a>	<a href="#">3</a>
<a href="#">2.3 Catalogues of OPTIONS (Options/*.py).....</a>	<a href="#">4</a>
<a href="#">2.4 Catalogues of type_element (Elements/*.py).....</a>	<a href="#">4</a>

## 1 Catalogues of orders

---

- As much as possible to use the possibilities of the supervisor concerning exclusions, the values by default...
- When an argument is of standard "text":  
S' it can take a finished number of values, to give the integral list of the possibilities by the keyword into.
- The comments are welcome.
- To make validate the vocabulary in RTA.
- To consult the document [D5.01.01].

## 2 Catalogues of elements

---

### 2.1 Catalogue `phenomenons_modelisations.py`

- The names of the phenomena and modelings must be validated by the EDA/RTA because they appear to the user.

### 2.2 Catalogue `physical_quantities.py`

- To give a name to the sizes of the form `s` where `s` can be worth:
  - R : reality
  - C : complex
  - F : function (K8)
- When one does not want to create a new too particular size, to use the "neutral" sizes: `NEUT_R` or `NEUT_K24`.
- When the catalogue of the SIZES is modified, to think of updating the document "description of the sizes" [D4.04.02] and to classify the names of sizes alphabetically.
- Not to define sizes of the type: L, K32, K80
- Not to destroy components `CMPS` in an existing size without to have checked that none `type_element` does not use it.
- Not to change the order of the `CMPS` of an existing size without modifying them `type_element` who use it.
- When one introduces a new component into a size, to put it following the existing `CMPS`. This avoids "breaking" programming too much "into hard"; for example, a programmer can have made:
  - checking that 'DX' and 'DY' are the first two `CMPS` of the size 'DEPL\_R',
  - then use of `DEPL_R (I), I = 1.2.`

## 2.3 Catalogues of OPTIONS (Options/\*.py)

- The names of options must be validated by the EDA/RTA if they appear to the user.
- Before adding a new parameter in an option, it should be looked at if there already does not exist in the catalogue of the shared parameters (`Commons/parameters.py`)
- Not to reinvent the names of the parameters for each option; to take as a starting point those already selected. The usual form is the following one: `nom_par = 'P' //nom_gd`.  
Examples: `PMATERF`, `PGEOMER`
- To comment on field parameter: example:  
`PCAGEPO = InputParameter (... comment= " RAY AND THICKNESS FOR the PIPES"`
- If the field which is associated with the parameter always the same "origin has". One can indicate his "container", i.e. the structure of data where this field is.  
For example:  
`PCAGNBA = InputParameter (... container=' CARA! .CARGENBA',`
- When a new option is added, it is necessary to fill the block " `CondCalcul` ". This task is not easy, because it requires to have a knowledge of all the elements. In the doubt, it is always necessary to prefer to declare a list of elements vaster than necessary. When a user is confronted with a blocking due to this excess of prudence, it will be always time to raise the restriction by adding a line "-" in the block.

## 2.4 Catalogues of type\_element (Elements/\*.py)

- Before adding a local new fashion in a catalogue of element, it should be looked at if there already does not exist in the catalogue of the shared local modes (`Commons/located_components.py`)
- For the names of the local modes to take as a starting point the names chosen by `type_element` neighbors. To respect the use:
  - Cxxxx: mode of type map
  - Nxxxx: mode of the `cham_no` type
  - Exxxx: mode of the `cham_elem` type
- To have doubts about the coherence of `type_element` that one modifies with the others  
`type_element` :
  - why the new one `type_element` would it have a local mode "with the nodes" whereas all the others have it "at the points of GAUSS"?
  - why the new one `type_element` doesn't it use this field parameter, this component?