% if one calculates the “vicinity” of the grid:

```
(F) '.VGE'          :  sd_voisinage
```

Structures of data `sd_maillage, sd_voisinage, sd_squelette` and `sd_grille`

Contents

1 General information

A grid (or `sd_maillage`) is a set of meshs of the predefined types: `HEXA8`, `TRIA3`,…

These meshs are defined by a list of nodes which have coordinates. These are the nodes which connect the meshs between them. The whole of the coordinates of the nodes of the grid forms a field with the nodes of the size "geometry" (`cham_no/GEOM_R`).

A grid can also contain groups of meshs and groups of named nodes. These groups are unspecified: a mesh (for example) can belong to $0,1,2,\ldots,n$ groups.

Let us announce that for the static under-structuring, a grid can contain super-meshs (meshs having an unspecified number of nodes).

When the grid is made of linear elements forming a continuous line, it can contain a map containing for each mesh the curvilinear X-coordinate of each node of the mesh.

A skeleton (or `sd_squelette`) is a grid of visualization of the results for dynamic substructures. It is a grid with some objects moreover.

A grid (or `sd_grille`) is a typical case of grid for which all the nodes are aligned according to the directions of a local base.

If the grid is obtained after a cutting of the Local type Average Contact (LAKE), then `sd_maillage` contains 3 additional objects : `.PATCH, .CONOPA, .COMAPA`.

2 Tree structure of the Structures of Data

```
sd_maillage (K8):: = record

                (O) '.DIME'      :    OJB      S      V       I
                    '.NOMNOE'    :    OJB      S      NR       K8
                    '.COORDO'    :    sd_cham_no (GEOM_R)

                (F) '.GROUPENO':    OJB      XD   V        I      NO ()
                (F) '.PTRNOMNOE':    OJB      S      NR       K24

                % if the grid contains meshs:
                |     (O) '.NOMMAI'    :  OJB      S      NR      K8
                          '.TYPMAIL'   :  OJB      S      V       I
                          '.CONNEX'    :  OJB      XC     V       I      NAKED ()

                      (F)  '.GROUPEMA': OJB      XD     V       I      NO ()
                      (F)  '.PTRNOMMAI': OJB      S      NR      K24

                % if the grid contains super-meshs (static under-structuring):
                |     (O) '.NOMACR'    :  OJB      S       V     K8
                          '.PARA_R'    :  OJB      S       V     R
                          '.SUPMAIL'   :  OJB      XD      V     I      NO ()
                      (F) '.TYPL'      :  OJB      S       V     I

                % if the grid were obtained by RESTRICTED CREA_MAILLAGE/ :
                |     (O) '.MAOR'   :  OJB      S       V     K8   long=1
                          '.CRNO'   :  OJB      S       V     I
                          '.CRMA'   :  OJB      S       V     I
```

```
% if the grid contains a curvilinear map of X-coordinate:
(F) '.ABS_CURV'    :  sd_carte (ABSC_R)


% if the grid were refined by Lobster:
(F) '.ADAPTATION'  :  OJB    S     V    I    LONG=1

% if the grid were read with the format 'MED' :
(F) '(11) .FORMES'    :  OJB    S     V    K32  LONG=2

% if the grid were obtained by CREA_MAILLAGE/DECOUPE_LAC :
|    (O) '.PATCH'        :  OJB      XC      V     I
         '.CONOPA'       :  OJB      S       V     I
         '.COMAPA'       :  OJB      S       V     I
         '.PTRNOMPAT'    :  OJB      S       V     K24
```

```
sd_voisinage (K12)       :: = record

        (O)  'PTVOIS'         :   OJB  V  I
        (O)  'ELVOIS'         :   OJB  V  I

sd_squelette (K8)        :: = record

        (O)  '$vide'          :   sd_maillage
        (F)  '.INV.SKELETON'  :   OJB  V  I
        (F)  '.CORRES'        :   OJB  V  I
        (F)  '(9) .NOMSST'     :   OJB  V  K24

        % if DEFI_SQUELETTE/GRID:
        (F)     '.ANGL_NAUT'  :   OJB  V  R   dim = 3
                '.TRANS'      :   OJB  V  R   dim = 3

sd_grille (K8)         :: = record

        (O)  '$vide'          :   sd_maillage
        (O)  '.GRLI'          :   OJB  V  I
        (O)  '.GRLR'          :   OJB  V  R
```

3 Contents of the objects jeveux

3.1 Object .DIME

```
'.DIME' : S  V  I    LENGTH = 6

   V (1): nb_no          : many physical nodes of the grid
   V (2): nb_nl          : many nodes of Lagrange of the grid
          nb_nl > 0      : there exist super-meshs: static under-structuring ( sss )
   V (3): nb_ma          : many meshs of the grid
   V (4): nb_sm          : many super-meshs of the grid
          nb_sm  > 0     : static under-structuring
   V (5): nb_sm_mx       : raising amongst super-meshs
   V (6): dim_coor       : /2  (if grid 2D)
                           /3  (if grid 3D)
```

3.2 Object .NOMNOE

'.NOMNOE' : S NR K8 LENGTH = nb_no

It is the pointer of names giving the correspondence:

nom_de_nœud ↔ numéro_de_nœud

3.3 Object .GROUPENO

'.GROUPENO' : XD V I NO () VARI NB_OJB = nb_gno

- number of group_no nb_gno = NUTIOC ('.GROUPENO')

That is to say V = '.GROUPENO' (nom_gno)

nb_no_gno = many nodes of nom_gno = LONUTI (V)

for I = 1, nb_no_gno

V (I): number of I ème node of nom_gno

Notice important :

One group_no can be empty (0 node). An empty group is represented by a vector of LONMAX=1 and LONUTI=0. When one wants to know the number of elements of a group, it is thus necessary to use JELIRA/LONUTI. When a group is created, it is necessary to think of informing (JEECRA) two attributes (LONMAX and LONUTI).

Caution :

The number of group_no a grid can change: one can modify a grid (order DEFI_GROUP) to add to him group_no.

3.4 Object .PTRNOMNOE

'. PTRNOMNOE ' : S NR LONG K24 = nb_grno

It is the external pointer of name being used to store the names of groups of nodes.

3.5 Object .NOMMAIL

'.NOMMAI' : S V LONG K8 = nb_ma

It is the pointer of names giving the correspondence:

nom_de_maille ↔ numero_de_maille

3.6 Object .TYPMAIL

'.TYPMAIL' : S V I LONGB = nb_ma

V (ima) : number of the type of mesh associated with the e-mailE of number ima

- the types of mesh are defined in the catalogue /compelem/type_maille__.cata :
  SEG2, TRIA3, QUAD4,..., HEXA20
- the types of mesh recognized by Aster are described in [U3.01]

- correspondence: `nom_de_type_de_maille` ↔ `numero_de_type_de_maille` is accessible by the pointer from name: `'&CATA.TM.NOMTM'` cf [D4.04.01].

3.7 Object .CONNEX

`'.CONNEX' : XC V I NAKED () NB_OJB = nb_ma`

That is to say `V = '.CONNEX' (ima)`

`V (1)`: number of the 1st node of the mesh of number: `ima`
…
`V (N)`: number of the last node of the mesh of number: `ima`

`N` = many nodes of the mesh `ima = LONMAX (V)`

The number of nodes of a mesh is always the number of nodes associated with the type of mesh who is attached to him (see object `'&CATA.TM.NBNO'` [D4.04.01])

3.8 Object .GROUPEMA

`'.GROUPEMA': XD V I NO () VARI NB_OJB = nb_gma`

- number of `group_ma nb_gma = NUTIOC ('.GROUPEMA')`

That is to say `V = '.GROUPEMA' (nom_gma)`

`nb_ma_gma` = many meshs of `nom_gma = LONUTI (V)`

for `I = 1, nb_ma_gma`

`V (I)`: number of Ième mesh of `nom_gma`

Notice important :

One `group_ma` can be empty (0 mesh). An empty group is represented by a vector of `LONMAX=1` and `LONUTI=0`. When one wants to know the number of elements of a group, it is thus necessary to use `JELIRA/LONUTI`. When a group is created, it is necessary to think of informing (`JEECRA`) two attributes (`LONMAX` and `LONUTI`).

Caution:

The number of `GROUP_MA` a grid can change: one can modify a grid (order `DEFI_GROUP`) to add to him `group_ma`.

3.9 Object .PTRNOMMAI

`'. PTRNOMMAI ' : S NR LONG K24 = nb_grma`

It is the external pointer of name being used to store the names of groups of meshs.

3.10 Objects .NOMACR, .PARA_R, .SUPMAIL and .TYPL

`'.NOMACR' : S V K8 LENGTH = nb_sm`

for `I = 1, nb_sm`

`V (I)` : name of `MACR_ELEM_STAT` associated with the super-mesh i

'.PARA_R' : `S V R LENGTH = 14*nb_sm`

for `I = 1, nb_sm:`

```
V (14* (i-1) +1): TX
V (14* (i-1) +2): TY
V (14* (i-1) +3): TZ
V (14* (i-1) +4): alpha
V (14* (i-1) +5): beta
V (14* (i-1) +6): gamma
V (14* (i-1) +7): PX
V (14* (i-1) +8): PY
V (14* (i-1) +9): PZ
V (14* (i-1) +13): dmini
V (14* (i-1) +14): dmaxi
```

- (`TX, TY, TZ`) are the values of translation of the geometrical transformation associated with the super-mesh `I`
- (`alpha, beta, gamma`) are the nautical angles (in radians) defining the rotation of the geometrical transformation,
- (`PX, PY, PZ`) defines the preceding centre of rotation.

That is to say `macrost` `MACR_ELEM_STAT` associated with the super-mesh `I`, the position of the super-mesh `I` is defined by isometry of the nodes of `macrost`. The isometry is the composition in the order: rotation THEN translation.

- `Dmini:` minimal distance between 2 nodes of the mesh `I`,
- `dmaxi:` maximum distance between 2 nodes of the mesh `I`.

'.SUPMAIL' : `XD V I NO () VARI NB_OJB = nb_sm`

That is to say `V = '.SUPMAIL' (nom_sma)`

`V` is a vector containing the numbers of the nodes of the super-mesh `nom_sma`. The nodes of a super-mesh can be of “physical” type or type “Lagrange”.

That is to say:
`inop` a “physical” number of node of the super-mesh `nom_sma`
`inol` a number of node “Lagrange” of the super-mesh `nom_sma`

$$1 \leq inop \leq nb\_no$$
$$nb\_no + 1 \leq inol \leq nb\_no + nb\_nl$$

`V` is the shape of recopy of the object `'.CONX'` `MACR_ELEM_STAT` [D4.08.01].

`V` the connectivity of the super-meshs defines.

The super-meshs “are restuck” by “physical” nodes.

Nodes of “Lagrange” inherited `MACR_ELEM_STAT` are never common to several super-meshs.

Full number of nodes (“physical” + “Lagrange”) of `nom_sma = LONMAX (V)`

The pointer of names (intern) of the object `'.SUPMAIL'` give the correspondence:

number (super_maille) ↔ name (super_maille)

'.TYPL' : S V I LENGTH = nb_nl

for I = 1, nb_nl
V (I): /-1 if the node of "Lagrange" I is of type "before"
/-2 if the node of "Lagrange" I is of type "after"

3.11 Object .ADAPTATION

'.ADAPTATION' : S V I LENGTH = 1

V (1): level of adaptation (by Lobster) grid.

This level is 0 in general. It is nonnull only if the grid were refined by Lobster. This information is used only by Lobster.

3.12 Object (11) .FORMES

'(11) .FORMES' : S V K32 LENGTH = 2

This object exists only if the grid were read with the format 'MED'
V (1): 'MED'.
V (2): name of the grid MED.

3.13 Objects .MAOR, .CRMA and .CRNO

These objects exist only in one grid obtained by the order:
RESTRICTED CREA_MAILLAGE/.

.MAOR (1): the name of the grid contains from which the grid (grid "origin") is resulting.

The object .CRMA give the correspondence between the numbers of the mailless of the grid origin and the restricted grid:
.CRMA (ima_re) - > ima_orig

The object .CRNO give the correspondence between the numbers of the nodes of the grid origin and the restricted grid:
.CRNO (ino_re) - > ino_orig

3.14 Objects .PTVOIS and .ELVOIS

These objects describe the neighbors of the meshs of a grid.

Each neighbor (i.e close mesh) has a type:

```
3D BY FACE    : F3: 1
2D BY FACE    : F2: 2
3D BY STOPS  : A3: 3
2D BY STOPS  : A2: 4
1D BY STOPS  : A1: 5
3D BY TOP  : S3: 6
2D BY TOP  : S2: 7
1D BY TOP  : S1: 8
```

```
0D BY TOP  : S0: 9
```

One stores in the objects `.PTVOIS` and `.ELVOIS` :

For any mesh `M0` grid:
  Full number of neighbors `NVTOT`
  For each neighbor `MV` :
    type of the neighbor
    number of the mesh
    many common tops enter `M0` and `MV: nso_com`
    For `is = 1, nso_com` :
      Local number in `M0`
      Local number in `MV`

3.15 Objects `'.PATCH', '.CONOPA', '.COMAPA'`

A macro-mesh is an object of the collection **'.PATCH'** and is characterized by 3 objects: **'.CONOPA', '.COMAPA'** and **'.PTRNOMPAT'**:

`'.PATCH'` : it is about a collection of whole objects contiguous with access by variable number of patch length

`'.CONOPA'` : it is about a vectors object of entireties making it possible to create the opposite connectivity of a node to its macro-mesh.

`'.COMAPA'` : it is about a vectors object of entireties making it possible to create the opposite connectivity of a mesh to its macro-mesh.

`'.PTRNOMPAT'` : it is about one vectors of K24 allowing to connect the number of the patch in the name of the associated group of mesh.

4 Specificity of the structure of data skeleton

A skeleton being grid of restitution of a calculation per under-structuring, the structure data skeleton is copied on that of the structure of data of a grid. One finds all the elements described there previously. One adds the object to it `'.INV.SKELETON'` in order to specify, for each node of the skeleton, the substructure of which he belongs in the beginning and to which node of this substructure he corresponds:

That is to say `nb_no` the number of nodes of the grid included in the skeleton.

`'.INV.SKELETON'` `:  OJB  V I    LENGTH = 2 * nb_no`

  That is to say `V = '.INV.SKELETON'`
    for `I = 1, nb_no`
      `V (I):` number of the substructure of origin of the node `I`
    for `I = nb_no + 1.2 * nb_no`
      `V (I):` number of node I in its substructure of origin

  If the skeleton were obtained starting from a skeleton `(DEFI_SQUELETTE/RECO_GLOBAL)` :

`'.CORRES'` `:  OJB  V I  LENGTH = nb_no-`**`nbmoi`**

C'is an optional object, it exists only lorsqu' one creates a skeleton by modifying old (keyword SKELETON is well informed in the catalogue of the order). In this case, one will amalgamate the nodes of the interfaces (all interfaces or only those given by the user) according to a criterion indicated behind the keyword CRITERION. One thus carries out the "sticking together". Here **nbmoi** is the number of nodes amalgamated with the dynamic interfaces (does not exist any more in the new skeleton).
For the node ino skeleton, .CORRES (ino) is the number of the node of the initial skeleton (before "sticking together").

'.NOMSST' : OJB V K8 LENGTH = nbss

Optional object which in the case of exists a skeleton known as "classical" (initial skeleton when one informs the substructures which will form it). Its length is equal to the number of the substructures which form the skeleton.
It contains the names of the substructures which form the skeleton.

Note:
This object is not useful apart from the order DEFI_SQUELETTE. Nevertheless, it must (unfortunately!) belong to sd_squelette. Indeed, if one makes:

```
SQUEL1= DEFI_SQUELETTE (...)
SQUEL2= DEFI_SQUELETTE (…, SQUELETTE=SQUEL1,…)
```

The object SQUEL1.NOMSST will be used to build SQUEL2.

If the skeleton were obtained starting from a grid (DEFI_SQUELETTE/GRID) :

'.ANGL_NAUT' : V LONG K8 = 3

nautical angles of the rotation of the skeleton compared to the grid from which it is resulting.

'.TRANS' : V LONG K8 = 3

coordinated vector which gives the translation of the skeleton compared to the grid from which it is resulting.

5 Specificity of the structure of given grid

The specificity of a grid is that its nodes are aligned according to the directions of a local base of the grid. This kind of grid is used with algorithms based on finite difference methods. Consequently, the local base $(X_{loc}, Y_{loc}, Z_{loc})$, the table of connection of the nodes and the distance between two aligned nodes of the grid must be stored by adding this information in sd_maillage.

That is to say nb_no the number of nodes of the grid which form the grate.

'.GRLI' : OJB V I LENGTH = 6*nb_no

In this object one stores the table of connection of the nodes of the grid.
That is to say V = '.GRLI':

for I = 1: nb_no
V (6* (i-1) +1) : node following the node i according to the direction of the vector X_{loc}

`V (6* (i-1) +2)` : preceding node the node i according to the direction of the vector X_{loc}
`V (6* (i-1) +3)` : node following the node i according to the direction of the vector Y_{loc}
`V (6* (i-1) +4)` : preceding node the node i according to the direction of the vector Y_{loc}
`V (6* (i-1) +5)` : node following the node i according to the direction of the vector Z_{loc}
`V (6* (i-1) +6)` : preceding node the node i according to the direction of the vector Z_{loc}

If it node i does not have following or preceding node in one of the three directions, the element corresponding of the vector V is put at zero. In this case the node i is on the free surface of the grid.

'.GRLR' : `OJB V R LENGTH = 10+6*nb_no`

In this object one stores the length of the smallest edge of the grid, the local base $(X_{loc}, Y_{loc}, Z_{loc})$ and outdistances it between the connected nodes of the grid.
That is to say $W='.GRLR'$

`W (1)` : length of the smallest edge of the grid

`W (2)` : component X first vector X_{loc} local base of the grid
`W (3)` : component Y first vector X_{loc} local base of the grid
`W (4)` : component Z first vector X_{loc} local base of the grid

`W (5)` : component X second vector Y_{loc} local base of the grid
`W (6)` : component Y second vector Y_{loc} local base of the grid
`W (7)` : component Z second vector Y_{loc} local base of the grid

`W (8)` : component X third vector Z_{loc} local base of the grid
`W (9)` : component Y third vector Z_{loc} local base of the grid
`W (10)` : component Z third vector Z_{loc} local base of the grid

for `I = 1: nb_no`
`W (10+6* (i-1) +1)` : absolute value of distance enters the node i and the node `V (6* (i-1) +1)` according to according to the direction of the vector X_{loc}
`W (10+6* (i-1) +2)` : absolute value of the distance enters the node i and the node `V (6* (i-1) +2)` precedent according to the direction of the vector X_{loc}

`W (10+6* (i-1) +3)` : absolute value of distance enters the node i and the node `V (6* (i-1) +3)` according to according to the direction of the vector Y_{loc}
`W (10+6* (i-1) +4)` : absolute value of the distance enters the node i and the node `V (6* (i-1) +4)` precedent according to the direction of the vector Y_{loc}

`W (10+6* (i-1) +5)` : absolute value of distance enters the node i and the node `V (6* (i-1) +5)` according to according to the direction of the vector Z_{loc}

`W (10+6* (i-1) +6)` : absolute value of the distance enters the node i and the node `V (6* (i-1) +6)` precedent according to the direction of the vector Z_{loc}

If the node i does not have following or preceding node in one of the three directions, the element corresponding of the vector W is put at zero. In this case the node i is on the free surface of the grid.