

**Manuel d'Utilisation**  
**Fascicule U4.1- : Gestion**  
**Document : U4.11.03**

## Procédure *POURSUIITE*

---

### 1 But

---

Poursuivre une étude à partir de la sauvegarde au format JEVEUX ou au format HDF de sa base 'GLOBALE'.

La syntaxe apparemment complexe de cette procédure ne doit pas inquiéter l'utilisateur, l'appel avec les opérandes par défaut, est suffisant dans la plupart des cas : *POURSUIITE* ( )

L'usage de cette commande est tout à fait semblable à celui de *DEBUT*.

## 2 Syntaxe

POURSUIITE

```
(
  ◇ PAR_LOT = / 'OUI' , [DEFAULT]
              / 'NON' ,

  ◇ IMPR_MACRO = / 'NON' , [DEFAULT]
                 / 'OUI' ,

  ◇ BASE = _F (
    ◆ FICHIER = / 'VOLATILE' ,
                / | LONG_ENRE= lenr, [I]
                  | NMAX_ENRE= nenr, [I]
                  | LONG_REPE= lrep, [I]
  ),
  ◇ CODE = _F (
    ◆ NOM = nom code, [K8]
    ◇ UNITE = | 15 , [DEFAULT]
              | unitd, [I]
  ),
  ◇ IMPRESSION = F (
    ◆ FICHIER = nomlocal [1_Kn]
    ◆ UNITE = uniti , [1_I]
  ),
  ◇ ERREUR = _F (
    ◇ ERREUR_F = / 'ABORT' , [DEFAULT]
                 / 'EXCEPTION' ,

  ◇ DEBUG = _F(
    ◇ JXVERI = / 'OUI' ,
               / 'NON' ,
    ◇ ENVIMA = 'TEST' , [1_Kn]
    ◇ JEVEUX = / 'OUI' ,
               / 'NON' ,
    ◇ SDVERI = / 'OUI' ,
               / 'NON' , [DEFAULT]
  ),
  ◇ MEMOIRE = _F(
    ◇ GESTION = / 'RAPIDE' , [DEFAULT]
                 / 'COMPACTE' ,
    ◇ TYPE_ALLOCATION = / ty, [I]
                      / 1, [DEFAULT]
    ◇ TAILLE = ta, [I]
    ◇ PARTITION = pa, [R]
    ◇ TAILLE_BLOC = / 800., [DEFAULT]
                   / tbloc, [R]
  ),
  ◇ RESERVE_CPU = _F(
    ◇ / VALE = vale [R]
      / POURCENTAGE = pcent [R]
    ◇ BORNE = / bv, [R]
              / 180. [DEFAULT]
  ),
  ◇ FORMAT_HDF = / 'NON' , [DEFAULT]
                 / 'OUI' ,
)
)
```

### 3 Principe de fonctionnement

Cette procédure affecte, en outre, les ressources mémoire nécessaires à la poursuite du calcul.

Les opérandes de la commande sont homologues de ceux de la procédure *DEBUT* [U4.11.01]. Ils permettent de préciser certaines ressources affectées à la nouvelle exécution.

L'étude engagée précédemment se poursuit par un ensemble de commandes commençant par *POURSUITE* et se terminant par *FIN* [U4.11.02].

Les commandes placées avant *POURSUITE* (sauf évidemment *DEBUT*) ou après *FIN*, si elles sont syntaxiquement correctes, sont ignorées.

La procédure *POURSUITE* qui est exécutée, dès sa lecture par le superviseur, effectue les tâches suivantes :

- définition des unités logiques des fichiers utilisés en impression,
- allocation des fichiers associés aux bases de données gérées par *JEVEUX*,
- lecture des catalogues de commandes mais pas des catalogues des éléments qui ont été recopiés sur la base de données lors de la première exécution.

Les opérandes sont à utiliser pour dérouter les différents fichiers sur des numéros d'unité logique différents des numéros affectés par défaut ou pour ajuster certains paramètres de fichiers.

Les concepts de python simples (de type variable) créés lors d'une exécution précédente sont conservés dans un fichier associé à la base *JEVEUX* (pick.1). lors de l'exécution de la procédure *POURSUITE* ces concepts sont régénérés et peuvent donc être utilisés sous le nom sous lequel ils ont été créés.

### 4 Opérandes

L'opérande *PAR\_LOT* et les mots clés *IMPRESSION* et *DEBUG* sont identiques à ceux de la procédure *DEBUT* [U4.11.01].

Le mot clé *BASE* est différent pour la procédure *POURSUITE*.

#### 4.1 Opérande *PAR\_LOT*

◇ *PAR\_LOT* =

Mode de traitement des commandes :

'OUI' : (option par défaut) ; le superviseur analyse **toutes** les commandes avant d'en demander l'exécution.

'NON' : après avoir analysé **une** commande le superviseur demande son exécution puis passe à l'analyse (et à l'exécution) de la commande suivante (traitement commande par commande).

#### 4.2 Mot clé *IMPR\_MACRO*

◇ *IMPR\_MACRO* =

Autorise ou non les affichages produits par les macros dans le fichier de message. La lecture des fichiers de message peut être pénible quand elle contient la totalité des échos des sous-commandes générées par la macro elle-même. Par défaut, seul l'écho des commandes explicitement appelées par l'utilisateur dans son jeu de commandes apparaîtra.

## 4.3 Mot clé BASE

◇ BASE =

La fonctionnalité de ce mot clé est de redéfinir les valeurs des paramètres des fichiers d'accès direct associés aux «base de données» dans le cas où l'on ne désire pas utiliser ceux fixés par défaut. La taille maximum du ou des fichiers associés, et par conséquent le nombre maximum d'enregistrements, peut être redéfini à l'aide du paramètre passé sur la ligne de commande derrière le mot clé

En mode *POURSUITE*, certaines caractéristiques de la base GLOBALE ne peuvent plus être modifiées.

Valeurs par défaut des paramètres associés aux bases de données

VOLATILE		
NMAX_ENRE	15728	
LONG_ENRE	100	K mots
LONG_REPE	2000	

Le mot vaut 8 octets sur plate-forme 64 bits sous LINUX 64, TRU64 et IRIX 64, 4 octets sur plate-forme 32 bits sous SOLARIS, HP-UX et WINDOWS-NT, LINUX.

Sous Linux 64, la procédure *POURSUITE*, avec les valeurs par défaut, allouera un fichier d'accès direct d'au plus 15728 enregistrements de 100 Kmots (le K vaut 1024) pour la base 'VOLATILE'.

### Remarques :

*La taille réelle du fichier est dynamique ; elle dépend du volume d'informations à stocker effectivement. Mais cette taille est limitée par les conditions d'exploitation et un paramètre défini parmi les valeurs caractérisant la plate-forme. Sur la plate-forme de référence Linux 64 la taille maximum est fixée à 12 Go. Cette valeur peut-être modifiée en passant un argument sur la ligne de commande de l'exécutable derrière le mot clé -max\_base taille où taille est une valeur réelle mesurée en Mo, ou dans les arguments de ASTK [U1.04.00].*

*Sur les plates-formes 32 bits, la taille maximum est fixée à 2.047 Go (2 147 483 647), mais le code gère plusieurs fichiers pour aller au delà de cette limite lorsque le paramètre -max\_base est passé en argument.*

*Pour la base Globale, qui peut être sauvegardée et ré-utilisée en donnée d'un calcul, la taille maximum en « POURSUITE » est conservée telle quelle si le paramètre -max\_base n'est pas utilisé, mais peut-être redéfini au besoin de cette manière.*

### 4.3.1 Opérande FICHIER

◆ FICHIER =

Seuls les paramètres des bases de données 'LOCALE' et 'VOLATILE' peuvent être redéfinis.

### 4.3.2 Opérandes LONG\_ENRE / NMAX\_ENRE / LONG\_REPE

Définition des paramètres de la base de données (fichiers d'accès direct).

/ | LONG\_ENRE = lenr

lenr est la longueur des enregistrements en Kmots des fichiers d'accès directs utilisés.

**Remarque :**

Le gestionnaire de mémoire JEVEUX utilise ce paramètre pour déterminer deux types d'objets : les gros objets qui seront découpés en autant d'enregistrements que nécessaire, et les petits objets qui seront accumulés dans un tampon de la taille d'un enregistrement avant d'être déchargé.

NMAX\_ENRE = nenr

nenr est le nombre d'enregistrements par défaut, cette valeur est déterminée à partir de LONG\_ENRE et d'un paramètre d'exploitation sous IRIX 64 fixé à 12 Go (12 884 901 888 octets) pour la taille maximale du fichier associé à une base de données.

**Remarque :**

Les deux opérandes LONG\_ENRE et NMAX\_ENRE doivent être utilisés avec précaution, un mauvais usage pouvant conduire à l'arrêt brutal du programme par saturation des fichiers d'accès direct. La cohérence entre la taille maximale du fichier et la valeur résultant du produit des deux paramètres LONG\_ENRE et NMAX\_ENRE est vérifiée en début d'exécution.

LONG\_REPE = lrep

lrep est la longueur initiale du répertoire (nombre maximal d'objets adressables par JEVEUX), elle est gérée dynamiquement par le gestionnaire de mémoire qui étend la taille du répertoire et de tous les objets système associés au fur et à mesure des besoins.

## 4.4 Mot clé IMPRESSION

◇ IMPRESSION =

Définition des unités logiques des fichiers utilisés en impression.

### 4.4.1 Opérande FICHIER

◆ FICHIER =

Liste de noms symboliques de fichiers.

### 4.4.2 Opérande UNITE

◆ UNITE = uniti

Numéro de l'unité logique associée aux fichiers de la liste.

Si uniti est négatif ou nul, il n'y a pas d'impression sur ce(s) fichier(s).

Par défaut :

FICHIER	UNITE
'ERREUR'	9
'MESSAGE'	6
'RESULTAT'	8
'MED'	80

La définition de l'association nom de fichier, numéro d'unité logique alimente la structure de données interne au code qui est généré par la commande `DEFI_FICHIER` [U4.12.03].

## 4.5 Mot clé ERREUR

◇ ERREUR =

Permet de récupérer une erreur de type <F> pour effectuer un traitement particulier, ce mécanisme a été mis en place pour vérifier l'émission de message d'erreur dans les tests de non-régression du code. Il est aussi intéressant de pouvoir récupérer proprement la main dans certaines macros (Stanley ou outils métiers) sans s'arrêter brutalement en erreur fatale.

### 4.5.1 Opérande ERREUR\_F

◇ ERREUR\_F =

'ABORT' le comportement du code est inchangé et le code s'arrête en imprimant une remontée d'erreur.

'EXCEPTION' on lève l'exception `aster.FatalError` (code 20) et on revient au comportement standard en cas d'erreur ('ABORT')

## 4.6 Mot clé DEBUG

◇ DEBUG =

Option de débogage (réservée aux développeurs et à la maintenance du code).

### 4.6.1 Opérande JXVERI

◇ JXVERI = 'OUI'  
'NON'

Permet de contrôler l'intégrité de la mémoire entre deux exécutions de commandes consécutives. Par défaut l'exécution s'effectue sans "DEBUG".

### 4.6.2 Opérande ENVIMA

◇ ENVIMA = 'TEST'

Permet d'imprimer dans le fichier 'RESULTAT' les valeurs des paramètres définis dans le progiciel ENVIMA caractérisant la machine [D6.01.01].

### 4.6.3 Opérande JEVEUX

◇ JEVEUX = 'OUI'  
'NON'

Permet d'activer le mode de fonctionnement en debug du gestionnaire de mémoire JEVEUX : déchargements sur disque non différés et affectation des segments de valeurs à une valeur indéfinie [D6.02.01].

### 4.6.4 Opérande SDVERI

◇ SDVERI = 'OUI'

*L'usage de ce mot clé est à destination des développeurs.*

*Ce mot clé déclenche la vérification des structures de données produites par les opérateurs. Il est utilisé dans le cadre des procédures de développement du code dans les tests de non régression.*

## 4.7 Mot clé MEMOIRE

Permet de modifier le mode de gestion de la mémoire. Lors de l'allocation en mémoire d'un segment de valeurs, il est possible soit d'effectuer une recherche de place en provoquant des déchargements sur disque (`GESTION = 'COMPACTE'`), ce qui permet d'utiliser moins d'espace mémoire mais aux prix de nombreux accès disque, soit de rechercher de façon prioritaire les zones libres ou correspondant à des accès en lecture seule (`GESTION = 'RAPIDE'`).

### 4.7.1 Opérande GESTION

`GESTION =`

- 'COMPACTE' : permet d'activer le mode d'allocation mémoire le plus économe en place totale
- 'RAPIDE' : permet d'activer le mode d'allocation mémoire privilégiant un accès rapide

### 4.7.2 Opérande TYPE\_ALLOCATION

`TYPE_ALLOCATION = ty`

- 1 : gestion standard de mémoire, on ne distingue pas les objets à allouer,
- 2 : les objets systèmes de collection sont alloués en fin de zone mémoire de façon à éviter d'éparpiller ces derniers et de trop fractionner les zones susceptibles d'accueillir de gros objets,
- 3 : même type d'allocation que précédemment, mais s'appliquant sur un critère de taille des objets,
- 4 : la zone mémoire est partitionnée en deux, une zone est réservée à l'allocation des petits objets.

### 4.7.3 Opérande TAILLE

`TAILLE` = taille en mots (unité d'adressage en entier) définissant les petits objets utilisée pour un type d'allocation 3 ou 4.

### 4.7.4 Opérande PARTITION

`PARTITION` = rapport entre la taille de la zone mémoire utilisée pour l'allocation des gros objets et la zone totale (dans le cas où `TYPE_ALLOCATION = 2, 3` ou `4`).

Cette zone est située en fin de segmentation, les objets systèmes propres à JEVEUX sont alloués dans la partition réservée aux "petits" objets.

#### Remarque :

*Si l'une des partitions est saturée, on revient à un mode de gestion standard de la mémoire (`TYPE_ALLOC = 1`).*

### 4.7.5 Opérande TAILLE\_BLOC

Ce mot clé, autrefois placé sous `SOLVEUR` dans les commandes globales, est utilisé pour définir la taille des blocs de la matrice.

◇ `TAILLE_BLOC =`

On peut choisir la taille des blocs de la matrice de rigidité (`tbloc`). Cette taille est donnée en kiloR8 (1 kiloR8 = 1024 réels). Ce paramètre influe sur le nombre d'opérations d'entrée / sortie et donc sur le temps d'assemblage et de résolution. Par défaut cette valeur est fixée à 800 kiloR8, soit 8 enregistrements par défaut sur le fichier d'accès direct associé à la base JEVEUX.

## 4.8 Mot-clé **RESERVE\_CPU**

Permet de réserver une part du temps CPU attribué au job pour terminer proprement l'exécution en cas d'arrêt par manque de temps CPU détecté par une commande Aster. Ce mécanisme n'est utile que dans le cas d'une exécution batch de Code\_Aster. La valeur de cette réserve peut être indiquée en valeur absolue ou bien sous forme d'un pourcentage du temps CPU total. Cette valeur est bornée par la valeur du mot clé BORNE.

### 4.8.1 Opérande **VALE**

Valeur exprimée en secondes soustraite au temps CPU total, sur lequel certaines commandes globales se basent pour arrêter proprement l'exécution.

### 4.8.2 Opérande **POURCENTAGE**

Pourcentage soustrait au temps CPU total, sur lequel certaines commandes globales se basent pour arrêter proprement l'exécution.

### 4.8.3 Opérande **BORNE**

Valeur maximale de la réserve de temps, valant par défaut 180 secondes.

## 4.9 Mot clé **FORMAT\_HDF**

◇ `FORMAT_HDF = 'OUI'`

Permet de relire une base GLOBALE sauvegardée dans un fichier au format HDF (cf commande `FIN [U4.11.02]`). La base est alors reconstruite à partir des objets JEVEUX stockés dans le fichier, ce fichier peut avoir été construit sur une plate-forme différente (système d'exploitation, plate-forme 32 ou 64 bits). Les caractéristiques de la base originale sont relues dans le fichier et la base est reconstruite à l'identique (on conserve par exemple la longueur des enregistrements).

Le fichier associé à la base GLOBALE au format HDF est nommé `bhdf.1` dans le répertoire d'exécution du code.

## 5 Exemple d'utilisation

L'utilisation standard de cette procédure est :

`POURSUIT ( )`

Pour faire une exécution avec une base 'VOLATILE' avec une longueur d'enregistrement de 400 Kmots :

`POURSUIT ( BASE = _F ( FICHIER = 'VOLATILE' , LONG_ENRE = 400 ) , )`

Les tests `yyy100a` et `yyy100b` illustrent l'utilisation de `RESERVE_CPU`

Les tests `forma04b`, `ssnv 156a`, `ssnv166b`, `yyy108`, illustrent l'utilisation de `MEMOIRE`.