Utilitaires d'impression de messages

Résumé :

On présente dans ce document les utilitaires d'émission de messages d'information, d'erreur ou d'alarme en fortran (ou en python) (routine U2MESG en particulier) ainsi que le mécanisme de levée d'exception (routine UTEXCM).

Copyright 2015 EDF R&D - Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Table des matières

Copyright 2015 EDF R&D - Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

1 Exemple de message

```
!---------------------------------------------------------!
! <A> <MODELISA4_9>                                       !
!                                                         !
!                                                         !
!   -> Phase de vérification du maillage : présence de    !
!      mailles aplaties.                                  !
!                                                         !
!   -> Risque & Conseil :                                 !
!      Vérifiez votre maillage. La présence de telles mailles!
!      peut conduire à des problèmes de convergence et nuire !
!      à la qualité des résultats.                        !
!                                                         !
!                                                         !
!                                                         !
! Ceci est une alarme. Si vous ne comprenez pas le sens de !
! cette alarme, vous pouvez obtenir des résultats         !
! inattendus                                              !
!---------------------------------------------------------!
```

2 Type de messages

Les unités logiques d'impression sont établies en début d'exécution du travail, par le superviseur, les impressions émises par les routines décrites dans les paragraphes suivants seront effectuées sur ces unités logiques sans possibilité de redirection.

Les messages émis seront dirigés uniquement en fonction de leur type :

Code	Type de message	Fichiers de sortie
F	Message d'erreur fatale, l'exécution s'arrête après l'impression du message.	MESSAGE ERREUR RESULTAT
E	Message d'erreur, l'exécution continue (un peu) cf.[§2.2].	MESSAGE ERREUR RESULTAT
A	Message d'alarme.	MESSAGE RESULTAT
I	Message d'information.	MESSAGE
Z	Levée d'exception récupérable en python	

2.1 Les messages de type F

Ce type de message est suivi d'un arrêt immédiat de l'application, il est utilisé dans le cadre de la détection d'erreur grave ne pouvant permettre la poursuite normale d'une commande Aster. L'émission d'un message d'erreur **F** provoque l'arrêt de l'exécution.

Par défaut pour les « vrais » utilisateurs (si le mot clé DEBUT / CODE n'est pas utilisé), l'exécution s'arrête après avoir validé les concepts qui peuvent l'être :

- les concepts produits par les commandes déjà exécutées

Copyright 2015 EDF R&D - Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- Le concept en cours de création si la commande l'a prévu (par exemple, pour la commande STAT_NON_LINE, les pas de temps déjà archivés sont validés).

En revanche, si le fichier de commande utilise DEBUT / CODE, le code s'arrête immédiatement et appelle la routine abort() afin de créer un « core file » utilisable par le débogueur post-mortem.

2.2 Les messages de type E

Ce type de message permet d'analyser une série d'erreurs avant l'arrêt du programme. Par exemple, l'analyse syntaxique du fichier de commandes par le Superviseur ou l'analyse du fichier de maillage par la commande LIRE_MAILLAGE.

L'émetteur d'un message de type **E** doit émettre un message de type **F** à la fin de son analyse.

2.3 Les messages de type A

Il ne faut pas abuser des messages d'alarme de type **A** qui peuvent inquiéter inutilement les utilisateurs. Le message doit être clair et comporter des pistes pour éviter cette alarme.

Le nombre de messages d'alarme est limité automatiquement à 5 messages successifs identiques.

Il est recommandé aux utilisateurs qui ont des messages de type A de "réparer" leur fichier de commandes pour les faire disparaitre.

2.4 Les messages de type I

Les messages **I** sont des messages d'information.

2.5 Les messages de type Z

Les messages **Z** sont des messages permettant le mécanisme des « exceptions ». Ce type de message n'est utilisé que dans les 2 routines utexcm et utexcp. et ne doivent pas l'être ailleurs ! Voir [§8] pour les exceptions en général.

3 Fonctionnement général

Nous partons d'un exemple pour présenter le mode de fonctionnement des utilitaires d'impression de message:

3.1 Exemple

Message imprimé

```
!---------------------------------------------------!
! <A> <CHATON_2>                                    !
!                                                   !
!    Le petit chaton est de couleur rose.           !
!                                                   !
! Ceci est une alarme. Si vous ne comprenez pas le sens !
! de cette alarme, vous pouvez obtenir des résultats    !
! inattendus                                        !
!---------------------------------------------------!
```


Copyright 2015 EDF R&D - Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Ce message utilisateur Le petit chaton est de couleur rose. est constitué de deux parties:

- A : le texte fixe Le petit chaton est de couleur .
- B : le texte variable rose qui représente le contenu d'une variable de type caractère.

Pour mettre en œuvre dans le fortran ou dans le python l'impression de ce message, il faut :

- décrire dans un catalogue python la partie fixe (A) du message, avec le format des variables à imprimer
- instrumenter le sous-programme fortran ou la fonction python pour provoquer l'impression du message.

Appel FORTRAN

```
SUBROUTINE CHATON(...,...)
...
...
IF(I.EQ.1) THEN
  VALK = 'ROSE'
  CALL U2MESK ('A','CHATON_2',1,VALK)
ENDIF
...
...
END
```

Les arguments du sous-programme U2MESK :

- 'A' : précise que l'on imprime un message de type alarme
- 'CHATON_2' : est l'identificateur du message : CHATON est le nom du catalogue de message chaton.py et le chiffre 2 indique que l'on prendra dans ce catalogue le message n°2.
- 1: on imprime une seule variable caractère
- VALK : variable caractère à imprimer.

Catalogue message: chaton.py

```
cata_msg = {

1 : _(u"""
    Le petit chat est bien de couleur verte Ouf!
 """),

2 : _(u"""
    Le petit chat est de couleur %(k1)s.
 """),

3 : _(u"""
   Le petit chat a %(i1)d pattes et %(i2)d yeux.
 """),

4 : _(u"""
   Le petit chat pèse %(r1)f kilogrammes.
 """),

5 : _(u"""
```


Copyright 2015 EDF R&D - Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
            ATTENTION: Votre chat est bizarre, il :
               - a plus de 4 pattes, il en a %(i1)d,
               - est de couleur %(k1)s et %(k2)s,
               - est trop gros, il pèse (%(r1)f kilogrammes.

             On arrête les frais, ce n'est pas un chat !!
           """),

            }
```

4 Utilitaires fortran

Nom	Fonction
U2MESS	Imprime un message simple
U2MESK	Imprime à la fois un message et des valeurs de type caractère
U2MESI	Imprime à la fois un message et des valeurs de type entière
U2MESR	Imprime à la fois un message et des valeurs de type réelle
U2MESG	Imprime à la fois un message, des valeurs de type caractère, entière et réel

4.1 Impression de messages simples

SUBROUTINE U2MESS(TYPE , IDMESS)

TYPE	IN	Type de message (E,F,A,I)
IDMESS	IN	identificateur du message

Exemple:

```
CALL U2MESS('A','CHATON_1')
```

Imprime le message suivant :

```
<A> <CHATON_1>
Le petit chat est de couleur verte Ouf !
```

4.2 Impression de messages et des valeurs de type caractère

SUBROUTINE U2MESK(TYPE , IDMESS , NK , VALK)

TYPE	IN	Type de message (E,F,A,I)
IDMESS	IN	Identificateur du message
NK	IN	Nombre paramètres de type caractère
VALK	IN	Valeurs des paramètres de type caractères

Exemple :

```
VALK = 'ROSE'
CALL U2MESK('A',CHATON_2', 1, VALK)''
```

Imprime le message suivant :

```
<A> <CHATON_2>
Le petit chat est de couleur rose.
```


Copyright 2015 EDF R&D - Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

4.3 Impression de messages et des valeurs de type entière

SUBROUTINE U2MESI(TYPE , IDMESS , NI , VALI)

TYPE	IN	Type de message (E,F,A,I)
IDMESS	IN	Identificateur du message
NI	IN	Nombre de paramètres de type entier
VALI	IN	Valeurs des paramètres de type entier

Exemple :

```
VALI (1) = 4
VALI (2) = 2
CALL U2MESI('A','CHATON_3', 2, VALI)''
```

Imprime le message suivant :

```
<A> <CHATON_3>
 Le petit chat a 4 pattes et 2 yeux.
```

4.4 Impression de messages et des valeurs de type réel

SUBROUTINE U2MESR(TYPE , IDMESS , NR , VALR)

TYPE	IN	Type de message (E,F,A,I)
IDMES	IN	Identificateur du message
NR	IN	Nombre paramètres de type réel
VALR	IN	Valeurs des paramètres de type réel

Exemple :

```
 VALR (1) = 130.
 CALL U2MESR('A','CHATON_4', 1, VALR)''
```

Imprime le message suivant :

```
<A> <CHATON_4>
Le petit chat pèse 130.0 kilogrammes.
```

4.5 Impression de messages des valeurs de type caractères, entières et réelles

SUBROUTINE U2MESG(TYPE , IDMESS , NK , VALK , NI , VALI , NR , VALR)

TYPE	IN	Type de message (E,F,A,I)
IDMESS	IN	Identificateur du message
NK	IN	Nombre de paramètres de type caractère
VALK	IN	Valeurs des paramètres de type caractère
NI	IN	Nombre de paramètres de type entier
VALI	IN	Valeurs des paramètres de type entier
NR	IN	Nombre de paramètres de type réel

Copyright 2015 EDF R&D - Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

VALR	IN	Valeurs des paramètres de type réel

Exemple :

```
VALK (1) = 'bleue'
VALK (2) = 'rose'
VALI = 5
VALR  = 130.
U2MESG('A','CHATON_5', 2, VALK, 1 , VALI, 1,VALR)''
```

Imprime le message suivant :

```
<E> <CHATON_5>
ATTENTION: Votre chat est bizarre, il :
  - a plus de 4 pattes, il en a 5 ,
  - est de couleur bleue et rose,
  - est trop gros, il pèse 130.0 kilogrammes.
```

On arrête les frais, ce n'est pas un chat !!

4.6 Impression d'un message en plusieurs « morceaux »

Il est parfois pratique d'émettre un message « par morceaux » c'est à dire d'appeler plusieurs fois la routine U2MESG. Ceci est en pricipe impossible pour les messages d'erreur fatale (F/E) car le code s'arrête dès le premier message !
Par ailleurs, on souhaite que le message complet apparaisse à l'utilisateur comme un message unique. En particulier on veut qu'il apparaisse dans un même cadre.

Pour cela, on dispose du mécanisme 'A+', 'F+', … Le principe est d'ajouter un '+' au type du message tant que le message n'est pas terminé. Le dernier message du groupe (sans le '+') termine le message.

On pourra par exemple écrire un message en 2 morceaux en écrivant :

```
CALL U2MESK('F+', 'FONCT0_11',1,NOMF)
CALL U2MESR('F',  'FONCT0_26',3,VALR)
```

Ce qui, associé au catalogue suivant :

```
11 : _(u"""
L'interpolation de la fonction '%(k1)s' n'est pas autorisée.
Le type d'interpolation de la fonction vaut 'NON'

  -> Risque & Conseil :
    Voir le mot-clé INTERPOL des commandes qui créent des fonctions.
"""),

26 : _(u"""
   abscisse demandée : %(r1)f
   intervalle trouvé : [%(r2)f, %(r3)f]
"""),
```

conduira à un message ressemblant à :

```
!---------------------------------------------------------------------!
! <F> <FONCT0_11>                                                     !
!                                                                     !
! L'interpolation de la fonction fonc3 n'est pas autorisée.           !
```


Copyright 2015 EDF R&D - Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
! Le type d'interpolation de la fonction vaut 'NON'                    !
!                                                                      !
!  -> Risque & Conseil :                                               !
!    Voir le mot-clé INTERPOL des commandes qui créent des fonctions.  !
!                                                                      !
!   abscisse demandée : 105.                                           !
!   intervalle trouvé : [0., 100.]                                     !
!----------------------------------------------------------------------!
```


Copyright 2015 EDF R&D - Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Remarques:

- Dans un groupe de messages produisant un message « par morceaux », le type des différents messages dot être le même (F/A/I).
- L'identifiant du message imprimé est celui du premier message du groupe.

5 Utilitaire Python

L'impression des messages dans les fichiers python est réalisée par la méthode UTMESS.

Nom	Fonction
UTMESS	Imprime à la fois un message, des valeurs de type caractères, entières et réelles

def UTMESS(type, idmess, valk, vali, valr)

type	IN	Type de message (E,F,A,I)
idmess	IN	identificateur du message
valk	IN	Valeurs des paramètres de type caractères
vali	IN	Valeurs des paramètres de type entier
vali	IN	Valeurs des paramètres de type réel

Contrairement au fortran, il n'y a qu'une fonction UTMESS car les arguments valk, vali et valr sont optionnels (cf. Python Tutorial, keyword arguments). La fonction de U2MESI est réalisée de cette manière :

```
UTMESS('A', 'MESSAGE_1', vali=(1, 2, 3))
```

6 Format d'impression dans les catalogues de message

Format	Fonction
%(kn)s	Format d'impression de la n^{ième} valeur de type caractère
%(in)d	Format d'impression de la n^{ième} valeur de type entière
%(rn)f	Format d'impression de la n^{ième} valeur de type réelle (on utilise parfois le format %g, %e, %F, %E, %G)

En savoir plus sur les codes de formattage des chaines : voir String formatting operations de la documentation Python.

Il est nécessaire que les messages soient des chaines « unicode » (notez le « u » devant """) car les messages contiennent des caractères non-ascii.
Quelques règles simples doivent être suivies afin de conserver un maximum d'homogénéité entre les quelques 6.000 messages :

- Faire des phrases, pas d'abréviations, pas de noms de routines, de structure de données qui sont éventuellement connus des développeurs mais pas des utilisateurs.
- Ne pas faire de « mise en page » : pas de saut de ligne superflu, éviter de former des colonnes, etc. car il sera difficile de les conserver lors de la traduction.

Copyright 2015 EDF R&D - Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

7 Exemples

Identificateur du message	Catalogue de message (chaton.py)	Appel fortran/python
-	cata_msg = {	-
CHATON_1	1 : _(u""" - Le petit Chat est bien de couleur verte Ouf! """),	→ Fortran CALL U2MESS ('A','CHATON_1') → Python UTMESS('A','CHATON_1')
CHATON_2	2 : _(u""" - Le petit chat est de couleur %(k1)s """),	→ Fortran CALL U2MESK ('I','CHATON_2',1,COUL) → Python UTMESS ('I','CHATON_2',valk=...)
CHATON_3	3 : _(u""" - Le petit chat a %(i1)d pattes et %(i2)d yeux. """),	→ Fortran VALI(I)= NBPATT VALI(2)= NBYEUX CALL U2MESI ('I','CHATON_3',2,VALI,) → Python UTMESS('I','CHATON_2',vali=...)
CHATON_4	4 : _(u""" - Le petit chat pèse %(r1)f kilogrammes. """),	→ Fortran VALR = POIDS CALL U2MESR ('I','CHATON_4',1,VALR,) → Python UTMESS('I','CHATON_4',valr=...)
CHATON_5	5 : _(u""" - ATTENTION: Votre chat est bizzare, il : → a plus de 4 pattes, il en a %(i1)d, → est de couleur %(k1)s et %(k2)s, → est trop gros, il pèse(%(r1)f kilogrammes. On arrête les frais, ce n'est pas un chat !! """),	→ Fortran VALI = CINQ VALK(1)='BLEU' VALK(2) = 'ROSE' VALR= 130. CALL U2MESG ('E','CHATON_4',2,VALK,2,VALI,1,VALR,) → Python UTMESS('E','CHATON_5',valk,vali,vakr)
-	}	-

Copyright 2015 EDF R&D - Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

8 Utilitaires de levée d'exception

En cours d'écriture

Les utilitaires UTEXCP et UTEXCM permettent de lever une exception python depuis le fortran et de la transmettre au jeu de commande. Cette exception peut être récupérée dans le fichier de commande (ou une macro-commande) par un bloc try/except.

Les exceptions étant des cas particuliers d'erreur <F>, si elles ne sont pas récupérées, le superviseur termine le calcul comme après une erreur <F> ordinaire (fermeture et recopie de la base). Seule l'exception FatalError échappe à la règle puisqu'elle équivaut à une erreur <F> (et encore on verra un peu plus loin que ce comportement est modifiable).

Le mécanisme d'exception permet d'essayer une commande puis de reprendre la main si celle-ci échoue en levant une exception particulière.

Ce fonctionnement est particulièrement utilisé dans les macro-commandes, soit pour se comporter différemment selon le contexte, soit pour émettre un message plus parlant à l'utilisateur que celui qui aurait été émis par une commande fille.

Remarque:

L'utilisation des exceptions et surtout des blocs try / except *n'a de sens qu'en mode* PAR_LOT='NON'

8.1 Mise en œuvre dans le source

On présente ici la mise en œuvre du mécanisme des exceptions dans le source fortran et python du point de vue du développeur. On abordera un peu plus bas les exceptions avec la vision "utilisateur" (§8.4).

Pour la description de ce qu'est une exception, les mécanismes de levée (raise) et d'interception (except), on se reportera à la Description des exceptions du Tutorial Python.

"Lever une exception" signifie émettre un signal avec (souvent) des éléments de contexte pour permettre au programme appelant de réagir ainsi :

- je comprends le signal et je prends telle décision en conséquence
- je ne comprends pas le signal et je m'arrête (en émettant un autre signal au programme appelant du niveau supérieur)
- j'ignore le signal et je laisse le programme appelant du niveau supérieur traiter le signal

Et ainsi de suite, sachant qu'au plus haut, c'est l'interpréteur Python lui-même qui prendra la décision d'interrompre l'exécution.

Syntaxiquement et schématiquement, le fonctionnement est le suivant :

```
try:

   ...
   Instructions à exécuter qui peuvent par exemple lever 3 exceptions
   différentes : UneExceptionParticulière, UneAutreException, EncoreUneAutre
   ...

except UneExceptionParticulière, arguments:

...
```


Copyright 2015 EDF R&D - Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

On exécutera ces instructions si l'exception "UneExceptionParticulière" est levée. On peut utiliser les "arguments" utilisés lors de la levée de l'exception.
Dans le cas le plus simple (et fréquent), "arguments" est un message qui peut être imprimé avec "print" ou utilisé comme chaîne de caractères avec "str(arguments)".

```
   ...

except UneAutreException, arguments2:

   ...
   Instructions à exécuter si l'exception UneAutreException est levée.
   ...
```

Supposons que le bloc d'instructions sous le try lève l'exception EncoreUneAutre (qui n'est pas interceptée par un bloc except), l'exécution s'arrête avec un message du type :

```
Traceback (most recent call last):
  File ".......", line 3, in <.......>   <<<  permet de dire où s'est
                                               produite l'exception
EncoreUneAutre: ......                   <<<  éléments de contexte
                                               (arguments) fournissant
                                               des détails sur les raisons
                                               de la levée d'exception
```

Pour les autres mécanismes, try/finally ou try/except/else, on se reportera à la documentation Python.

8.1.1 Levée d'exception dans Code_Aster

Le plus simple : en Python, c'est-à-dire dans le fichier de commandes ou dans une macro-commande, il suffit de faire :

```
if nbre_iterations > nbre_maxi_autorise:
   raise aster.NonConvergenceError, "Absence de convergence avec le
nombre d'itérations autorisé"
```

Dans le source fortran (exemple dans nmerro.f qui traite les erreurs dans STAT_NON_LINE) :

```
      ...
      ELSE IF (ITEMAX) THEN
        CALL UTEXCP(22,'MECANONLINE_83')
      ...
```

où

- 22 est le numéro de l'exception
- 'MECANONLINE_83' est l'identifiant du message à imprimer

En effet, les exceptions propres à Code_Aster sont numérotées pour être accessibles facilement depuis le fortran :

numéro de l'exception	Description	Nom de l'exception dans le module aster	Type erreur
20	Erreur fatale	FatalError	F
21	Erreur non fatale	error	F
22	Non convergence	NonConvergenceError	F
23	Echec intégration du comportement	EchecComportementError	F

Copyright 2015 EDF R&D - Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

24	Bande de fréquence vide	BandeFrequenceVideError	F
25	Matrice singulière	MatriceSinguliereError	F
26	Echec de traitement du contact	TraitementContactError	F
27	Matrice de contact non inversible	MatriceContactSinguliereError	F
28	Manque de temps CPU	ArretCPUError	F
29	Echec du pilotage	PilotageError	F

Remarques

Les exceptions sont définies dans le module aster (astermodule.c). Ainsi, on y accède par aster.error .
A part FatalError , toutes les exceptions dérivent de error qui représente donc l'enveloppe des erreurs <F> .
Le comportement en cas d'erreur fatale (FatalError) est modifiable par l'utilisateur [U1.03.01] , ce choix est déterminé dans les commande DEBUT/POURSUITE [U4.11.01],[U4.11.03] mot-clé facteur ERREUR_F . Dans les macro-commandes, on utilisera la fonction aster.onFatalError(arg) pour modifier ce comportement.

8.2 Levée d'une exception avec impression d'un message

CALL UTEXCP(CODE, IDMESS)

CODE	IN	Numéro de l'exception
IDMESS	IN	Identificateur du message

Exemple :

```
CALL UTEXCP(23,'CHATON_7')
```

8.3 Levée d'exception avec impression de messages et de valeurs de type caractères, entières et réelles

CALL UTEXCM(CODE, IDMESS, NK, VALK, NI, VALI, NR, VALR)

CODE	IN	numéro de l'exception
IDMESS	IN	Identificateur du message
NK	IN	Nombre de paramètres de type caractère
VALK	IN	Valeurs des paramètres de type caractère
NI	IN	Nombre de paramètres de type entier
VALI	IN	Valeurs des paramètres de type entier
NR	IN	Nombre de paramètres de type réel
VALR	IN	Valeurs des paramètres de type réel

Exemple :

```
   VALI (1) = NDIM1
```


Copyright 2015 EDF R&D - Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
VALI (2) = NDIM2
VALR     = 5.D0
CALL UTEXCM(24,'CHATON_12', 0,' ',2, VALI,1,VALR)
```

8.4 Exemple d'utilisation

Dans cet exemple, on souhaite :

- en cas de non-convergence de STAT_NON_LINE, relancer les calculs en augmentant le nombre d'itérations,
- arrêter les calculs si une autre erreur se produit.

Cette utilisation est illustrée dans le cas-test ssnp125a.

Programmation

On lève l'exception NonConvergenceError qui porte le numéro 22 :

```
...
ELSE IF ((.NOT.CONVER) .AND. ITEMAX .AND. (.NOT.ARRET)) THEN
    ITAB(1) = NUMORD
    ITAB(2) = ITERAT
    CALL UTEXCM(22,'MECANONLINE_85', 0, K8B, 2, ITAB, 0, RTAB)
END IF
```

Fichier de commande

```
try :
     STATNL=STAT_NON_LINE(...)

except aster.NonConvergenceError, message:
     # non convergence
     print "on continue en augmentant le nombre d'itérations"
     STATNL=STAT_NON_LINE(reuse=STATNL,
                          ...
                          CONVERGENCE=_F(ITER_GLOB_MAXI=400,)
                          …)
```

Fichier message : mecanonline.py

```
85 : _(u"""
   Arret : absence de convergence au numéro d'instant : %(i1)d
                                   lors de l'itération : %(i2)d
"""),
```

Message imprimé

```
!---------------------------------------------------------------------!
! <EXCEPTION> <MECANONLINE_85>                                        !
!                                                                     !
!                                                                     !
!    Arret : absence de convergence au numéro d'instant : 123         !
!                                    lors de l'itération :  51        !
!                                                                     !
!                                                                     !
!---------------------------------------------------------------------!
```


Copyright 2015 EDF R&D - Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)