SOURCES DE LA FONCTION GROUPAGE FG10b

AVERTISSEMENT

L'objet de la diffusion de ces sources est de permettre une intégration plus rapide pour les établissements et pour les sociétés de services, d'une fonction groupage identique à celle utilisée dans GENRSA et dans AGRAF.

Ainsi la documentation de ces sources n'a pas vocation à faciliter les modifications du moteur de groupage, et elle a été délibérément rédigée sans souci d'explicitations exhaustives : il s'agit d'une documentation au niveau fonctionnel uniquement. Pour chacune des fonctions utilisées, on trouvera une description de ce qu'elle exécute, sans entrer dans les détails de sa réalisation.

La description de ces fonctions ainsi que des variables globales se trouve dans les fichiers de *header*. Il s'agit de sources écrites en langage C ANSI, compilables dans les environnements UNIX, DOS et Windows.

L'intégration de ces sources dans un logiciel est soumise à la signature préalable d'un contrat d'utilisation. La conformité de leur mise en œuvre peut-être validée par la comparaison du groupage obtenu avec celui de l'exécutable fourni avec les sources de la fonction groupage nommé fg10mainv10b.exe.

L'ATIH ne saurait être tenu responsable de modifications effectuées par d'autres qu'elle sur ces sources.

PRÉSENTATION GÉNÉRALE

La fonction groupage est monomodulaire, c'est-à-dire qu'elle s'utilise en n'appelant qu'une seule fonction, la fonction *grp_10b()*. Celle-ci se charge de toutes les opérations nécessaires au groupage d'un RSS. Ces opérations sont : la lecture des RUM, le contrôle des RUM, la détermination du séjour principal, le groupage proprement dit et la production des éléments nécessaire à la valorisation.

Pour réaliser ce dernier point de manière à ce que le calcul des suppléments effectué par la fonction groupage soit identique à celui réalisé par les logiciels GENRSA et AGRAF, une nouvelle fonction nommée *CtrlFicUM()* a été intégrée dans ce package. Elle permet d'ajouter aux tables de groupage existantes un fichier d'autorisation d'unités médicales dans un contexte sécurisé. La mise en œuvre de cette nouvelle table n'est pas obligatoire.

Cette documentation présente la manière d'utiliser les fonctions *grp_10b()* et *CtrlFicUM()*.

Nouveautés apportées par la Fonction Groupage 10b:

- **Gestion des tables "multipériodes"**

Chaque table binaire contient désormais plusieurs sous-tables correspondant chacune à une période donnée, tout en ne constituant qu'un seul fichier physique. La fonction groupage 10b utilise la bonne sous-table en fonction de la date de sortie du RSS traité. Les fichiers des tables binaires doivent se situer dans le répertoire des tables passé en paramètre à *grp_10b*.

- **Le nombre de RUM maximum par RSS est désormais 99 au lieu de 25**

- **Gestion des autorisations d'unités médicales**

Pour calculer certains GHS ainsi que plusieurs autres variables (notamment les suppléments réanimation, surveillance continue, soins intensifs, néonatologie…), la fg10b peut utiliser l'information sur le type d'autorisation contenue dans le RUM, ou peut dorénavant utiliser une information plus fine contenue dans un fichier d'autorisation d'UM dont la réalisation est décrite plus bas. Pour cela, une nouvelle variable est passée en paramètre à la fonction groupage, nommée "utiliseficum", indiquant l'option choisie par l'utilisateur. Cette variable "utiliseficum" est membre de la structure "GRP_ENTREE_10" passée en paramètre d'entrée à la fg 10b.
Si le choix est fait de se baser sur le fichier d'autorisation d'UM, la fonction groupage fg10b tente d'ouvrir le fichier nommé "ficum.txt" (le nom est figé) situé obligatoirement dans le répertoire des tables passé en paramètre. Si ce fichier n'existe pas, la fg 10b renvoie l'erreur 09 d'absence de fichier. **Ce fichier doit être obligatoirement généré par la fonction *CtrlFicUM()* décrite plus bas. Si ce n'est pas le cas, la fonction groupage le détectera grâce à l'analyse de la signature située à la dernière ligne du fichier. Elle renverra alors le code erreur correspondant.**
Si l'utilisateur indique à la fonction groupage de ne pas utiliser de fichier d'autorisation d'UM, alors celle-ci considère que c'est l'autorisation indiquée dans le RUM qui fait foi. Ainsi, l'unité médicale est autorisée durant tout le séjour délimité par les dates d'entrée et de sortie du RUM et pour l'autorisation inscrit dans ce RUM.

Comment générer le fichier "ficum.txt"?
C'est le rôle de la fonction *CtrlFicUM()* implémentée dans le fichier "CTRLGENFICUM.C" et dont le prototype est déclaré dans le fichier "CTRLGENFICUM.H". Ces 2 fichiers ne font pas partie des sources de la fonction groupage V10b en elle-même. Ils sont compilables de façon totalement indépendante des fichiers sources de la fonction groupage avec lesquels ils sont fournis.

La fonction *CtrlFicUM()* permet de générer à partir d'un fichier d'autorisation d'UM au format "brut" (c'est-à-dire non vérifié et/ou non trié) un nouveau fichier d'autorisation d'UM dans un format utilisable par la fonction groupage grp_10b et dans lequel figure une signature attestant que le fichier a été trié et vérifié par cette fonction *CtrlFicUM()*.

Syntaxe d'appel de la fonction *CtrlFicUM()*

```
cr=CtrlFicUM(char *nficumin, char *nreptabfg, int bgenerelog)

où :
nficumin : nom et chemin complet du fichier d'UM à contrôler et transformer
nreptabfg : nom et chemin complet du répertoire des tables de la fonction
groupage dans lequel sera généré le fichier "ficum.txt" (finissant par /)
int bgenerelog : différent de zéro si on veut qu'un fichier de log soit généré
dans le même répertoire que le fichier source nficumin, égal à zéro sinon.
```

Rappel important :
La fonction groupage grp_10b utilise le fichier d'UM nommé "ficum.txt" situé dans le répertoire des tables binaires. Donc il faut veiller à utiliser la fonction *CtrlFicUM* de façon à ce qu'elle génère le fichier "ficum.txt" dans ce répertoire. Autrement dit, il faut toujours passer en 2ème paramètre à CtrlFicUM : "<répertoire des tables> ".

Voici les tâches effectuées par *CtrlFicUM*:

- vérification du format du fichier d'UM en entrée (toutes les lignes doivent contenir 17 caractères strictement).
- Vérification des doublons (deux autorisations d'une même unité médicale ne peuvent pas avoir la même date de début d'autorisation).
- Calcul des dates de fin d'autorisation (égal à la date de début de l'autorisation suivante moins 1 jour).
- Calcul de la signature
- Génération du fichier d'UM en sortie

Format d'entrée du fichier d'autorisation d'UM à fournir à la fonction *CtrlFicUM()* :
Chaque ligne de ce fichier texte a le format suivant dans l'ordre:
*numéro de l'unité médicale sur 4 caractères
*autorisation de l'unité médicale sur 2 caractères
*date de début de l'autorisation sur 8 caractères
*nombre de lits sur 3 caractères

Format du fichier d'autorisation d'UM "ficum.txt" utilisé par la fonction groupage grp_10b et généré par la fonction *CtrlFicUM()*:
Chaque ligne de ce fichier texte a le format suivant dans l'ordre:
*numéro de l'unité médicale sur 4 caractères
*autorisation de l'unité médicale sur 2 caractères

*date de début de l'autorisation sur 8 caractères
*nombre de lits sur 3 caractères
*date de fin de l'autorisation sur 8 caractères
A la dernière ligne de ce fichier figure la signature du fichier sur 10 caractères.
Ce fichier est trié par numéro d'UM, autorisation d'UM et par date de début d'autorisation.

Valeur du code retour de la fonction *CtrlFicUM*:
La fonction renvoie zéro si tout s'est bien passé, et un code erreur de 1 à 5 sinon:
cr = 1 : impossible d'ouvrir le fichier d'UM source passé en paramètre d'entrée
cr = 2 : impossible de créer le fichier d'UM cible passé en paramètre de sortie
cr = 3 : impossible de créer le fichier log
cr = 4 : le format du fichier d'UM source n'est pas bon, ou certaines date de début d'autorisation ne sont pas au bon format ou sont incohérentes.
cr = 5 : au moins 2 dates de début d'autorisation pour une UM donnée sont identiques
cr = 6 : erreur système (erreur allocation mémoire)

Type de données et structures en paramètres de la fonction groupage 10b

La fonction groupage 10b contient 4 paramètres dont seul le premier est à renseigner par l'utilisateur, les 3 autres étant calculés et remplis par la fonction groupage elle-même. Tous ces paramètres ont pour type des structures décrites ci-après :

1. Le type GRP_ENTREE_10 :

Cette structure est utilisée pour rassembler les paramètres d'entrée de la fonction groupage.
Elle seul contient l'ensemble des paramètres d'entrée à passer à la fonction groupage.

```
typedef struct {
    char *rum2[tab_max];
    char dirtab[BUFSIZ];
    int ns;
    int type_etab;
    int utiliseficum;

} GRP_ENTREE_10;
```

`*rum2[tab_max]` Tableau de pointeurs sur des chaînes de caractères. Ces chaînes contiennent les différents RUM d'un RSS. Chaque chaîne (correspondant à un RUM) doit se terminer par un caractère nul ('\0'). Ce tableau comporte au plus **tab_max** pointeurs, **soit 99 éléments**. Les formats acceptés sont les suivants : 011.

`dirtab[BUFSIZ]` Chaîne de caractères contenant le chemin d'accès absolu, complet, pour atteindre le répertoire contenant les tables. Il doit être écrit de telle sorte que l'accès à une table donnée consiste à concaténer cette chaîne avec le nom de la table recherchée.

`int ns` Entier contenant le nombre de séjours élémentaires du malade au cours de son séjour dans l'établissement hospitalier. Un appel à la fonction *grp_10b()* avec ce paramètre à 0 entraîne la fermeture des tables de groupage.

`int type_etab`
Entier indiquant si l'établissement est ex DGF (=1), ou ex OQN (=2).

`int utiliseficum`
Entier égal à 1 si la fonction groupage doit utiliser le fichier d'UM "ficum.txt" présent dans le répertoire des tables binaires pour les autorisations des unités médicales. Egal à zéro si c'est l'autorisation indiquée dans le RUM qui fait foi (dans ce cas, la fonction groupage considère que l'unité médicale est autorisée pour ce type d'UM durant tout le séjour dans l'UM).
Important : le fichier "ficum.txt" doit être généré par la fonction CtrlFicUM() décrite plus haut.

2. Le type GRP_RESULTAT_10 :

Cette structure est calculée et renvoyée par la fonction groupage afin de fournir les résultats traditionnels du groupage comme le GHM.

```
typedef struct {
      int ns_sor;
      int sejp;
      char version[3];
      char cmd[3];
      char ghm[5];
      char cret[4];
      int ErrCtrl[MAXERRCTRL];
      int ErrArb[MAXERRARB];
      int ErrImpl[MAXERRIMPL];
} GRP_RESULTAT_10 ;
```

`int ns_sor` Entier contenant le nombre de séjours après le traitement dans la fonction groupage. Ce nombre est désormais égal au nombre de séjours en entrée.

`int sejp` entier, destinée à recevoir le numéro du RUM désigné par le groupage comme contenant le diagnostic principal et le diagnostic relié, en débutant la numérotation des RUM à 0.

`char version[3]` Chaîne de 3 caractères destinée à recevoir le numéro de version de la classification lu dans les tables.

`char cmd[3]` Chaîne de 3 caractères de long, destinée à recevoir le numéro de la CMD déterminée par le groupage (terminée par le caractère NULL).

`char ghm[5]` Chaîne de 5 caractères de long, destinée à recevoir le numéro du GHM déterminé par le groupage (terminée par le caractère NULL).

`char cret[4]` Chaîne de 4 caractères de long, destinée à recevoir le code retour déterminé par le groupage. Il s'agit de la valeur en caractère du retour de la fonction *grp_10b()* (cf. plus bas). Cette chaîne est terminée par le caractère NULL.

`int ErrCtrl[MAXERRCTRL]` Tableau de **1+MAXERRCTRL** (101) entiers courts correspondant au vecteur des erreurs contenues dans chacun des RUM du RSS Il est construit de la façon suivante :

entier n°1	: (nr) nombre de RUM détectés dans le RSS
entier n°2	: (n1) nombre d'erreurs détectées dans le RUM n°1
…	
entier n°(nr+1)	: nombre d'erreurs détectées dans le RUM n°nr
entiers n°(nr+2) à (nr+1+n1)	: codes des erreurs du RUM n°1
entiers n°(nr+n1+2) à (nr+1+n1+n2)	: codes des erreurs du RUM n°2
…	

`int ErrArb[MAXERRARB]` Tableau de **1 + MAXERRARB** (6) entiers courts correspondant au vecteur des erreurs détectées par l'arbre de décision de la classification. Les erreurs sont stockées dans ce vecteur de manière séquentielle. Ce vecteur n'a pas de structure particulière.

`int ErrImpl[MAXERRIMPL]` Tableau de **1 + MAXERRIMPL** (21) entiers courts correspondant au vecteur des erreurs dues à l'implémentation. Les erreurs sont stockées dans ce vecteur de manière séquentielle. Ce vecteur n'a pas de structure particulière.

3. Le type GHS_VALORISATION_10

Cette structure est calculée et renvoyée par la fonction groupage afin de fournir les éléments de valorisation du séjour :

```
typedef struct {
    char ghs[5];
    int nbjxsup;
    int sejxinf;

    int nbjrea;
    int nbrep
    int nbjsrc;
    int nbjstf_issuderea;
    int nbjstf_tot;
    int nbjsra;
    int nbjssc;

    int nbac687;
    int nbac688;
    int nbac689;
    int acprlvtorg;

    int pih_a;
    int pih_b;

    int nbjnna;
    int nbjnnb;
    int nbjnnc;

    char prestdialavsros[5];

    int nbhd;
    int nbent1;
    int nbent2;
```

```
    int nbent3;
    int nbchhypb;
    int nbseance;
    int nbseanceavantsros;

 } GVAL GHS_VALORISATION_10;
```

`char ghs[5]` Chaîne de 5 caractères de long, destinée à recevoir le numéro du GHS déterminé à partir du GHM obtenu par le groupage, et des informations contenues dans le fichier ghsinfo.tab. Dans le cas de la dialyse pour les établissements ex-OQN, cette variable contient le libellé du forfait dialyse après SROS (par exemple D09,D10,D11,D14,D15,D16…). Pour les établissements ex-DGF dans le cas de la dialyse, elle contient le numéro de GHS après SROS (par exemple 9500,9501,9502,9520,9521,9522).

`int nbjxsup` Entier contenant le nombre de journées au-delà de la borne extrême haute pour le GHS considéré.

`int sejxinf` Entier utilisé comme booléen, contenant 0 si la durée du séjour n'est pas inférieure à la borne extrême basse du GHS ou 1 sinon.

`int nbjrea` Entier contenant le nombre de journées de réanimation après SROS.

`int nbrep` Entier contenant le nombre de journées de réanimation pédiatrique.

`int nbjsrc` Entier contenant le nombre de journées de surveillance continue après SROS.

`int nbjstf_issuderea`
Entier contenant le nombre de journées de soins intensifs provenant de la réanimation.

`int nbjstf_tot`
Entier contenant le nombre de journées de soins intensifs total, y compris les journées de soins intensifs provenant de la réanimation.

`int nbjsra` Entier contenant le nombre de journées de réanimation avant SROS (=0 si ex DGF).

`int nbjssc` Entier contenant le nombre de journées de surveillance continue avant SROS (=0 si ex DGF).

`int nbac687` Entier contenant le nombre d'actes supplémentaires (hors séances) de radiothérapie menant dans le GHM 28Z11Z.

`int nbac688` Entier contenant le nombre d'actes supplémentaires (hors séances) de radiothérapie menant dans le GHM 28Z12Z.

`int nbac689` Entier contenant le nombre d'actes supplémentaires (hors séances) de radiothérapie menant dans le GHM 28Z13Z.

`int acprlvtorg` Entier contenant la catégorie de prélèvement d'organes du séjour considéré. (0, 1, 2 ou 3).

int pih_a Entier utilisé comme booléen, contenant 0 en cas d'absence de code diagnostic associé Z75.80, 1 sinon.

int pih_b Entier utilisé comme booléen, contenant 0 si le mode d'entrée et le mode de sortie est égal à 0, 1 sinon.

int nbjnna Entier contenant le nombre de journées de néonatologie de type 2A.

int nbjnnb Entier contenant le nombre de journées de néonatologie de type 2B, y compris les journées de type 3 sans actes de réanimation et les journées dans une unité de réanimation pédiatrique sans actes de réanimation.

int nbjnnc Entier contenant le nombre de journées de néonatologie de type 3 avec actes de réanimation, ajouté du nombre de journées de réanimation pédiatrique avec actes de réanimation.

int prestdialavsros

Entier contenant : pour les établissements ex-DGF, contient le numéro du GHS avant SROS (9500,9501,9502,9503…) associé au GHM de dialyse obtenu . Pour les établissements ex-OQN, contient le libellé du forfait dialyse avant SROS (D01,D03,D04,D05…) associé au GHM obtenu.

int nbhd Entier contenant le nombre d'actes supplémentaires hors séances pour l'hémodialyse.

int ent1 Entier contenant le nombre d'actes supplémentaires hors séances pour les entraînements à la dialyse péritonéale automatisée.

int ent2 Entier contenant le nombre d'actes supplémentaires hors séances pour les entraînements à la dialyse péritonéale continue ambulatoire.

int ent3 Entier contenant le nombre d'actes supplémentaires hors séances pour les entraînements à l'hémodialyse.

int nbchhypb

Entier contenant le nombre de caissons hyperbare.

int nbseance Entier contenant le nombre de séance total.

int nbseanceavantsros

Entier contenant le nombre de séances avant SROS.

4. Le type RSS_INFO_10

Cette structure est calculée et renvoyée par la fonction groupage afin de fournir des éléments d'informations concernant le RSS, et également des informations propres à chaque RUM du RSS.

```
typedef struct {
          int nbdiag;
          char *listdiag;
     } DIAGLIST_10;
typedef struct {
          int nbac;
          char *listzac;
     } ACTLIST_10;
```

```
typedef struct {
          int brea;
          char typum[2];
          int dsp;
     } INFO_RUM_10;

typedef struct {
     int agea;
     int agej;
     int dstot;
     DIAGLIST_10 diaglist ;
     ACTLIST_10 actlist;
     INFO_RUM_10 tabval[tab_max];
     char sigfg[5];
     char sigrss[20];
     char sigrssghs[8];
}RSSI RSS_INFO_10;
```

`int agea` Entier contenant l'âge en années utilisé par la fonction groupage

`int agej` Entier contenant l'âge en jours utilisé par la fonction groupage

`int dstot` Entier contenant la durée totale de séjour calculée par la fonction groupage

`DIAGLIST_10 diaglist` Est une structure contenant :

- le nombre de diagnostics différents..
- la liste de ces diagnostics avec en première position le diagnostic principal, puis en deuxième position le diagnostic relié, puis l'ensemble des diagnostics associés retenus pour le RSS.

`ACTLIST_10 actlist` Est une structure contenant :

- le nombre de zones d'actes du RSS.
- la liste des zones d'actes .

La présentation de ces zones d'actes est identique à celle du RUM

`INFO_RUM_10 tabval[tab_max]` est un tableau de tab_max éléments contenant pour chaque RUM le type d'autorisation du dernier jour passé dans l'unité médicale (variable typum), la durée de séjour partielle (variable dsp) ainsi qu'un booléen indiquant la présence de journée de réanimation (Actes marqueurs+IGS >= borne, variable brea).

`char sigfg[5]`
Chaîne de 5 caractères contenant la signature de la fonction groupage utilisée pour grouper le RSS

`char sigrss[20]`
Chaîne de 20 caractères contenant la signature du RSS

`char sigrssghs[8]`
Chaîne de 8 caractères contenant la signature du GHS obtenu et de la variable sigrss.

Syntaxe de l'appel de la fonction *grp_10b()*

```
Seul le paramètre grp_ent_10 doit être renseigné avec l'appel:

cr=grp_10b(grp_ent_10,&grp_res_10,&ghs_val_10,&rss_inf_10)

où :
GRP_ENTREE_10 grp_ent_10;
GRP_RESULTAT_10 grp_res_10;
GHS_VALORISATION_10 ghs_val_10;
RSS_INFO_10 rss_inf_10;
```

Valeur en retour (*code-retour)*

La fonction *grp_10b()* renvoie un entier égal à zéro si les contrôles n'ont rien détecté d'anormal, et si le groupage n'a pas trouvé d'anomalies. Dans le cas contraire, la valeur retournée est le numéro de l'erreur détectée (sauf pour certaines erreurs de contrôles non bloquantes non retournées par la fonction groupage).

Ce code-retour est également placé, sous forme d'une chaîne, dans la structure GRP_RESULTAT_10 passée en deuxième paramètre à la fonction *grp_10b()*.

Toute erreur de contrôle dont la valeur est contenue dans la liste erreursbloq_10 du fichier fg10b.h est irrémédiable, et ne permet pas d'obtenir le groupage du R.S.S. Les erreurs de contrôle non contenues dans cette liste ne sont pas graves en terme de groupage (par exemple : date système improbable). De telles erreurs ne sont détectées que par l'analyse du vecteur décrit plus haut (ErrCtrl).
Les erreurs d'implémentation (vecteur ErrImpl) sont aussi irrémédiables, ainsi que les erreurs de groupage (tableau ErrArb), excepté pour l'erreur de groupage 80.

Compte tenu de la numérotation particulière des erreurs (trois catégories) l'exploitation du code retour seul nécessite de connaître le résultat du « groupage » : 90Z03Z pour une erreur d'implémentation bloquante, 90Z00Z pour un contrôle bloquant, 90C01Z, 90Z01Z ou 90Z02Z pour une erreur bloquante dans l'arbre décisionnel (erreur de groupage), tout autre groupe ou GHM pour un résultat correct (et dans ce cas, code-retour nul ou ayant une valeur non contenue dans la liste erreursbloq_10).

Exemple d'utilisation de la fonction groupage 10b

Les fichiers sources de la fonction groupage FG10b sont constitués des 15 fichiers suivants (hors tables binaires):
CCAM10.C
CCAM10.H
CONTROLE10.C
CONTROLE10.H
ERREURS10.C
ERREURS10.H
FG10B.C
FG10B.H
GRPSTRUCT10.H
RUM10.H
RUMFMT10.H

SELECTEU10.C
TABLES10.H
GESTIONUM.C
GESTIONUM.H

Afin de fournir 1 exemple d'utilisation de la fonction groupage, il est également fourni avec ce "package" 2 autres fichiers sources de démonstration nommés :

FG10MAINV10B.C
FG10MAINV10B.H

- **Compilation**

Les 2 fichiers FG10MAINV10B.C et FG10MAINV10B.H permettent de grouper un fichier de RSS (au format 011), en appelant la fonction groupage V10b pour chacun de ces RSS.
Le fichier exécutable nommé FG10MAINV10B.EXE fourni est le résultat de la compilation des fichiers sources comprenant :

- les sources de la FG10 b
- les 2 fichiers FG10MAINV10B.C ET FG10MAINV10B.H

- **Utilisation du fichier exécutable "FG10MAINV10B.EXE"**

Cet exécutable permet de grouper un fichier de RSS avec la fonction groupage FG10b. Voici la syntaxe de l'appel :

FG10MAINV10B.EXE {nom du fichier de RSS} {répertoire des tables} {type d'établissement} {utilise_ficum}

Le nom du fichier de RSS est le chemin complet du fichier de RSS à grouper.
Le répertoire des tables doit finir par le caractère '/'. Dans ce répertoire, les fichiers des tables binaires doivent s'y trouver.
Le type d'établissement doit être égal à 1 pour un établissement ex-DGF et à 2 pour un établissement ex-OQN.
Le paramètre utilise_ficum est égal à 1 si la fonction groupage doit utiliser le fichier d'UM "ficum.txt" présent dans le répertoire des tables binaires pour les autorisations des unités médicales. Egal à zéro si c'est l'autorisation indiquée dans les RUM qui fait foi (dans ce cas, la fonction groupage considère que l'unité médicale est autorisée pour ce type d'UM durant tout le séjour dans l'UM).

Exemple d'utilisation de la fonction *CtrlFicUM()*

Un exemple d'utilisation de cette fonction est implémenté à l'aide du fichier suivant : CTRLGENFICUM _MAIN.C.

- **Compilation**

Le fichier exécutable nommé " CTRLGENFICUM.EXE" est le résultat de la compilation des fichiers sources suivants:

- CTRLGENFICUM.C
- CTRLGENFICUM.H
- CTRLGENFICUM _MAIN.C

- **Utilisation du fichier exécutable " CTRLGENFICUM.EXE "**

Cet exécutable permet de générer un fichier d'UM dans un format reconnu par la fonction groupage V10b, à partir d'un format "brut" (non vérifié et/ou non trié). Voici la syntaxe de l'appel:

CTRLGENFICUM.EXE {nom complet du fichier d'UM source} {chemin complet du répertoire des tables} {fichier log}

Les paramètres de cet exécutable sont exactement les mêmes que ceux de la fonction *CtrlFicUM()* utilisée (voir plus haut pour les détails):
Les 2 premiers paramètres sont le fichier d'autorisation d'UM source et le répertoire dans lequel sera généré le fichier "ficum.txt" (finissant par /), et le 3 ème est différent de zéro si on souhaite qu'un fichier log soit générer au même endroit que le fichier source, égal à zéro dans le cas contraire. Ce fichier log permet de connaître plus précisément l'origine du problème lorsque la fonction *CtrlFicUM()* renvoie un code erreur.