



<p>Manuel d'utilisation Bibliothèques standards</p>

Table des matières

Librairie de date et heure NSDate	11
Installation	11
Extensions du langage NCL pour les dates et heures	11
Dates.....	11
Jours	11
Mois.....	12
Heures.....	12
Fichier NSDate.NCL	12
Catégories fonctionnelles de la librairie NSDate	13
Récupérer la date, le jour, l'heure courante	13
Choisir le format de date et d'heure.....	14
Récupérer et paramétrer les détails d'un format	18
Récupérer et paramétrer les noms de jour et de mois	24
Vérifier un format	31
Convertir de et vers des entiers et chaînes de caractères.....	31
Année jalon	34
Gestion d'erreurs (constantes retournées)	35
Librairie mathématique NSMath	37
Installation	37
Extensions du langage NCL pour les calculs mathématiques	37
Fichier NSMATH.NCL.....	37
Catégories fonctionnelles de la librairie NSMath	37
Partie entière et décimale d'un nombre réel	37
Carré, racine carrée et puissances.....	39
Constante "Pi"	41
Fonctions trigonométriques	42
Logarithmes et exponentielles.....	44
Librairie NSFexe	47
Installation	47

Extensions du langage NCL pour l'exécution de programmes	47
Fichier NSFEXE.NCL	47
Catégories fonctionnelles de la librairie NSFexe	48
Ouvrir et fermer des consoles DOS ou OS/2	48
Modes d'ouverture de console et de démarrage des programmes (constantes)	51
Démarrer / arrêter un programme sur une console	52
Démarrer / arrêter un programme.....	55
Librairie de fonctions diverses NSMisc.....	63
Installation	63
Fichier NSMISC.NCL	63
Catégories fonctionnelles de la librairie NSMisc	63
Démarrage d'une autre application.....	63
Environnement DOS.....	70
Gestion simple de fichiers.....	73
Gestion de répertoires et de disques.....	85
Gestion d'erreur.....	96
Gestion de fichiers texte.....	96
Gestion de fichiers binaires	108
Manipulation de contrôles et fenêtres	123
MDI Window Management	144
Recherche d'un fichier dans une liste de répertoires.....	148
Gestion des List Box	149
Manipulation des listes d'images et affectation d'images aux items de menu	160
Gestion de repertoires temporaires	173
Gestion des traces.....	175
Gestion de l'ancrage.....	179
Tri des tableaux	181
Autres	182
Codes d'erreur NSMisc.....	219
Liste des codes d'erreur NSMisc.....	219
Formats d'affichage	221

Formats de dates	221
Formats d'heures	222
Formats et spécifications.....	222
A propos des formats pour STRING\$ et ESTRING\$.....	225
Exemples	226
Librairie NSDynstr.....	229
Installation	229
Fichier NSDynstr.NCL	229
Référence de la librairie NSDynstr	229
Fonction IsDSNull% (Librairie NSDynstr)	229
Instruction SetDSNull (Librairie NSDynstr)	230
Fonction ForceDSLength (Librairie NSDynstr).....	230
Exemple de sauvegarde du contenu d'une chaîne dans un fichier (Librairie NSDynstr) ..	232
Librairie NSMLE	235
Installation (Librairie NSMLE)	235
Fichier NSMLE.NCL (Librairie NSMLE).....	235
Référence de la librairie NSMLE.....	235
Instruction NSMLE_CUT (Librairie NSMLE).....	235
Instruction NSMLE_COPY (Librairie NSMLE).....	236
Instruction NSMLE_PASTE (Librairie NSMLE)	236
Instruction NSMLE_CLEAR (Librairie NSMLE)	236
Fonction NSMLE_LOAD% (Librairie NSMLE)	236
Fonction NSMLE_SETTEXT% (Librairie NSMLE).....	237
Instruction NSMLE_GETTEXT (Librairie NSMLE).....	237
Fonction NSMLE_GETTEXTSIZE% (Librairie NSMLE)	238
Fonction NSMLE_ISSELECTED% (Librairie NSMLE)	238
Fonction NSMLE_ISUNDO% (Librairie NSMLE)	238
Fonction NSMLE_ISCLIPBOARD% (Librairie NSMLE)	239
Instruction NSMLE_UNDO (Librairie NSMLE)	239
Fonction NSMLE_SELECTION\$ (Librairie NSMLE).....	239
Fonction NSMLE_SEARCH% (Librairie NSMLE)	240

Fonction NSMLE_CHANGE% (Bibliothèque NSMLE)	240
Bibliothèque NSDYNCTRL.....	243
Installation et utilisation	243
Référence de la bibliothèque NSDYNCTRL	243
Constantes DCA_EF_* (Bibliothèque NSDYNCTRL)	243
Constantes DCA_LB_* (Bibliothèque NSDYNCTRL)	245
Constantes DCA_MLE_* (Bibliothèque NSDYNCTRL)	246
Constantes DCA_CB_* (Bibliothèque NSDYNCTRL).....	247
Constantes DCA_CHK_* (Bibliothèque NSDYNCTRL)	248
Constantes DCA_RB_* (Bibliothèque NSDYNCTRL).....	248
Constantes DCA_ST_* (Bibliothèque NSDYNCTRL)	249
Constantes DCA_BMP_* (Bibliothèque NSDYNCTRL)	250
Constantes DCA_GB_* (Bibliothèque NSDYNCTRL)	251
Constantes DCA_VS_* (Bibliothèque NSDYNCTRL).....	251
Constantes DCA_HS_* (Bibliothèque NSDYNCTRL).....	252
Constantes DCA_MNU_* (Bibliothèque NSDYNCTRL).....	252
Constante DI_WINCMNCTRL% (Bibliothèque NSDYNCTRL).....	252
Constante DI_WCC_Animation% (Bibliothèque NSDYNCTRL).....	254
Constante DI_WCC_DateTimePicker% (Bibliothèque NSDYNCTRL)	254
Constante DI_WCC_Hotkey% (Bibliothèque NSDYNCTRL)	256
Constante DI_WCC_ListView% (Bibliothèque NSDYNCTRL).....	256
Constante DI_WCC_MonthCalendar% (Bibliothèque NSDYNCTRL).....	258
Constante DI_WCC_ProgressBar% (Bibliothèque NSDYNCTRL)	259
Constante DI_WCC_TrackBar% (Bibliothèque NSDYNCTRL)	259
Constante DI_WCC_TreeView% (Bibliothèque NSDYNCTRL)	260
SEGMENT DC_ControlProperties (Bibliothèque NSDYNCTRL)	262
Fonction ItExists% (Bibliothèque NSDYNCTRL)	266
Fonction GetTotalDynamicControls% (Bibliothèque NSDYNCTRL).....	266
Fonction DC_CreateControl% (Bibliothèque NSDYNCTRL)	267
Fonction DC_DestroyControl% (Bibliothèque NSDYNCTRL)	267
Instruction DC_GetControlPropertiesFromID (Bibliothèque NSDYNCTRL).....	268

Instruction DC_GetControlProperties (Librairie NSDYNCTRL).....	269
Librairie NSMAPS.....	271
Référence de la librairie NSMAPS	271
Référence de la librairie NSMAPS.....	271
Instruction NSMAPS_MAP_REMOVE (Librairie NSMAPS).....	272
Fonction NSMAPS_MAP_ITERATOR_GETVALUE (Librairie NSMAPS)	273
Fonction NSMAPS_MAP_ITERATOR_GETSTRING (Librairie NSMAPS)	274
Fonction NSMAPS_MAP_GET_ITERATOR (Librairie NSMAPS)	274
Fonction NSMAPS_MAP_GET (Librairie NSMAPS)	275
Fonction NSMAPS_MAP_GET_STRING (Librairie NSMAPS)	276
Instruction NSMAPS_MAP_DISPOSE (Librairie NSMAPS)	277
Fonction NSMAPS_MAP_ITERATOR_NEXT (Librairie NSMAPS)	278
Fonction NSMAPS_MAP_NEW (Librairie NSMAPS).....	278
Fonction NSMAPS_MAP_ITERATOR_GETKEY (Librairie NSMAPS).....	279
Instruction NSMAPS_MAP_ITERATOR_DISPOSE (Librairie NSMAPS)	279
Fonction NSMAPS_MAP_SIZE (Librairie NSMAPS)	280
Fonction NSMAPS_MAP_NEW2 (Librairie NSMAPS).....	280
Instruction NSMAPS_MAP_PUT_STRING (Librairie NSJSON)	281
Fonction NSMAPS_MAP_STRING_NEW (Librairie NSMAPS).....	282
Instruction NSMAPS_MAP_PUT (Librairie NSMAPS).....	282
Fonction NSMAPS_MAP_CONTAINS% (Librairie NSMAPS).....	283
Exemple NSMAPS (Librairie NSMAPS)	284
Librairie nsDynCollec	291
Référence de la librairie NSDYNCOLLEC	291
Exemple DYNCOLLEC (Librairie NSDYNCOLLEC)	293
Fonction Da_append% (Librairie NSDYNCOLLEC)	295
Fonction DA_capacity% (Librairie NSDYNCOLLEC).....	295
Fonction DA_clear% (Librairie NSDYNCOLLEC).....	296
Fonction DA_CONTAINS% (Librairie NSDYNCOLLEC)	296
Fonction DA_CREATE (Librairie NSDYNCOLLEC)	297
Fonction DA_CreateWithCapacity (Librairie NSDYNCOLLEC).....	298

Fonction DA_delete% (Bibliothèque NSDYNCOLLEC)	298
Fonction Da_getAt (Bibliothèque NSDYNCOLLEC)	299
Fonction Da_getFirst (Bibliothèque NSDYNCOLLEC)	299
Fonction Da_getLast (Bibliothèque NSDYNCOLLEC)	299
Fonction Da_indexOf% (Bibliothèque NSDYNCOLLEC)	300
Fonction DA_insertArrayAt% (Bibliothèque NSDYNCOLLEC)	300
Fonction DA_InsertAt% (Bibliothèque NSDYNCOLLEC)	301
Fonction DA_insertDynArrayAt% (Bibliothèque NSDYNCOLLEC)	302
Fonction DA_isEmpty% (Bibliothèque NSDYNCOLLEC)	302
Fonction DA_LastIndexOf% (Bibliothèque NSDYNCOLLEC)	303
Fonction DA_remove% (Bibliothèque NSDYNCOLLEC)	303
Fonction DA_removeAt% (Bibliothèque NSDYNCOLLEC)	304
Fonction Da_reserve% (Bibliothèque NSDYNCOLLEC)	304
Fonction DA_setAt (Bibliothèque NSDYNCOLLEC)	305
Fonction DA_size% (Bibliothèque NSDYNCOLLEC)	305
instruction DA_sort (Bibliothèque NSDYNCOLLEC)	306
Bibliothèque de gestion de LOG nclLOGR	309
Référence de la bibliothèque nclLOGR	310
Référence de la bibliothèque nclLOGR	310
Instruction nsLogTrace (Bibliothèque NCLLOGR)	310
Instruction nsLogDebug (Bibliothèque NCLLOGR)	311
Instruction nsLogInfo (Bibliothèque NCLLOGR)	311
Instruction nsLogInfo (Bibliothèque NCLLOGR)	312
Instruction nsLogWarn (Bibliothèque NCLLOGR)	313
Instruction nsLogError (Bibliothèque NCLLOGR)	313
Instruction nsLogFatal (Bibliothèque NCLLOGR)	314
Fonction NSIsLogLevelEnabled% (Bibliothèque NnlLogr)	314
Constantes_LL_* (Bibliothèque nclLogger)	315
Exemple nclLOGR (Bibliothèque nclLOGR)	316
Compatibilité avec l'ancienne bibliothèque NSLOGR.NCL	317
Bibliothèque NSCRYPT	319

Référence de la librairie nsCrypt.....	319
Référence de la librairie nsCrypt	319
Fonction NS_HASH_STR\$ (Librairie NsCrypt)	319
Fonction NS_HASH_PKCS5_PBKDF2_HMAC\$ (Librairie NsCrypt).....	321
Fonction NS_GET_HASH_ERROR% (Librairie NsCrypt)	322
Fonction NS_GET_HASH_ERROR_MESSAGE\$ (Librairie NsCrypt)	323
Constantes NSCRYPT (Librairie nsCrypt)	323
Exemple nsCrypt (Librairie nsCrypt)	325
Glossaire.....	327
Index	331

LIBRAIRIE DE DATE ET HEURE NSDate

Cette librairie permet de gérer les dates et heures (date et heure courante, format des heures et des dates, nom des jours et des mois, etc.)

Installation

Déclarez NSDate.NCL dans les librairies nécessaires au développement de votre application.

Vérifiez que le fichier NSxxDATE.DLL est bien placé dans un des répertoires du PATH sous Windows.

Vérifiez que la date et l'heure de la machine sont correctes (commandes DATE et TIME de DOS).

Extensions du langage NCL pour les dates et heures

Dates

Le format de date est, par défaut, celui spécifié dans le panneau de configuration (Control Panel) de la machine d'exécution.

Vous pouvez modifier ce format par SETDATEFORMAT, qui accepte les formats :

DD MM YY%, MM DD YY%, YY MM DD%, DD MM YYYY%, MM DD YYYY%, YYYY MM DD%.

Le séparateur utilisé entre mois, jour et année ("- " ou "/" par exemple), peut être modifié par SETDATESEPARATOR. De plus, GETDATEFORMAT% et GETDATESEPARATOR\$ retournent le format utilisé.

La date initiale de référence est le 1 Janvier 1900. La librairie NSDate permet toutefois de gérer les dates "négatives", antérieures à 1900, depuis le 1 Janvier 1583.

CURRENTDATE% indique la date courante de la machine sur laquelle cette instruction est en train d'être exécutée, en nombre de jours depuis la date initiale de référence.

DATE\$ et DATE% convertissent un entier en date et réciproquement. L'entier est exprimé en nombre de jours depuis la date initiale de référence, ce nombre étant négatif pour les dates antérieures au 1/1/1900. La constante DATE_ERROR% est la valeur retournée par DATE% en cas d'erreur.

ISDATE% teste si une chaîne est bien au format d'une date.

Jours

La fonction CURRENTDAY% retourne le jour de la semaine, 0 correspondant au dimanche.

Les instructions SETSHORTDAY / SETLONGDAY et les fonctions GETSHORTDAY\$ / GETLONGDAY\$ permettent d'associer aisément les entiers 0 à 6 avec des chaînes représentant en clair le jour correspondant, de dimanche à samedi.

Mois

Les instructions SETSHORTMONTH / SETLONGMONTH et les fonctions GETSHORTMONTH\$ / GETLONGMONTH\$ permettent d'associer aisément les entiers 0 à 11 avec des chaînes représentant en clair le mois correspondant, de janvier à décembre.

Heures

Le format d'heure est par défaut celui spécifié dans le Panneau de Configuration. Ce format peut être modifié par SETTIMEFORMAT qui accepte les formats :

HH MM%, HH MM AMPM%, HH MM SS%, HH MM SS AMPM%

Le séparateur utilisé entre heures, minutes, et secondes (":" par exemple) est modifiable par SETTIMESEPARATOR. De plus, GETTIMEFORMAT% et GETTIMESEPARATOR\$ retournent le format utilisé.

Dans le cas des formats HH_MM_AMPM% et HH_MM_SS_AMPM% (voir HH MM*%), les chaînes affichées pour différencier matin et après-midi sont modifiables par SETTIMEAMPM. De plus, GETTIMEAM\$ et GETTIMEPM\$ retournent les chaînes utilisées.

L'heure initiale de référence est 0 heure, 0 minute, 0 seconde.

CURRENTTIME% indique l'heure courante de la machine sur laquelle cette instruction est en train d'être exécutée, en nombre de secondes depuis l'heure initiale de référence.

TIME\$ et TIME% convertissent un entier en heure et réciproquement. L'entier est exprimé en nombre de secondes depuis l'heure initiale de référence. La constante TIME ERROR% est la valeur retournée par TIME% en cas d'erreur.

ISTIME% teste si une chaîne est bien au format d'une heure.

Fichier NSDATE.NCL

Les verbes de la librairie NSDate, décrits ci-après, sont déclarés dans le fichier texte écrit en NCL, de nom NSDATE.NCL. Ce fichier peut aussi contenir des verbes complémentaires (API non publique). Vous pouvez donc désirer consulter ce fichier pour avoir la référence exhaustive de la librairie.

Pour consulter le fichier NSDATE.NCL :

1. Placez-vous dans le répertoire <NATSTAR>\NCL ou <NSDK>\NCL.
<NATSTAR> représente le répertoire que vous avez choisi au moment de l'installation de NatStar et <NSDK> celui de NS-DK.
2. Editez le fichier NSDATE.NCL avec n'importe quel éditeur de texte.

Catégories fonctionnelles de la librairie NSDate

Récupérer la date, le jour, l'heure courante

Fonction CURRENTDATE% (Librairie NSDate)

Retourne la date courante.

Syntaxe	CURRENTDATE%
Valeur retournée	INT(4) contient le nombre de jours écoulés depuis le 01/01/1900.

Exemple :

```
MESSAGE "The current date is:", DATE$(CURRENTDATE%)
```

Voir aussi CURRENTTIME%, DATE\$, DATE%

Fonction CURRENTDAY% (Librairie NSDate)

Retourne le jour courant de la semaine.

Syntaxe	CURRENTDAY%
Valeur retournée	INT(1) comprise entre 0 (dimanche) et 6 (samedi).

Exemple :

```
MESSAGE "Aujourd'hui, on est le : ", GETLONGDAY$(CURRENTDAY%)
```

Voir aussi GETSHORTDAY\$, GETLONGDAY\$

Fonction CURRENTTIME% (Librairie NSDate)

Retourne l'heure courante.

Syntaxe	CURRENTTIME%
Valeur retournée	INT(4) contient le nombre de secondes depuis 00:00:00.

Exemple :

```
MESSAGE "L'heure actuelle est:", TIME$(CURRENTTIME%)
```

Voir aussi CURRENTDATE%, TIME\$, TIME%

Choisir le format de date et d'heure

Constantes DD_MM_YY*% (Librairie NSDate)

Valeurs logiques employées pour définir le format de date.

Syntaxe	Déclaration interne	Description
DD_MM_YY%	1	Format sous forme <Jour>, <Mois>, <Année sur 2 chiffres>
DD_MM_YYYY%	4	Format sous forme <Jour>, <Mois>, <Année sur 4 chiffres>

Ces valeurs sont employées par SETDATEFORMAT et retournées par GETDATEFORMAT%.

Deux autres formats sont possibles :

- MM_DD_YY% ou MM_DD_YYYY%
- YY_MM_DD% ou YYYY_MM_DD%

Voir aussi MM DD YY*%, YY* MM DD%

Constantes MM_DD_YY*% (Librairie NSDate)

Valeurs logiques employées pour définir le format de date.

Syntaxe	Déclaration interne	Description
MM_DD_YY%	2	Format sous forme <Mois>, <Jour>, <Année sur 2 chiffres>
MM_DD_YYYY%	5	Format sous forme <Mois>, <Jour>, <Année sur 4 chiffres>

Ces valeurs sont employées par SETDATEFORMAT et retournées par GETDATEFORMAT%. Deux autres formats sont possibles :

- DD_MM_YY% ou DD_MM_YYYY%
- YY_MM_DD% ou YYYY_MM_DD%

Voir aussi DD MM YY*%, YY* MM DD%

Constantes YY*_MM_DD% (Librairie NSDate)

Valeurs logiques employées pour définir le format de date.

Syntaxe	Déclaration interne	Description
YY_MM_DD%	3	Format sous forme <Année sur 2 chiffres>, <Mois>, <Jour>
YYYY_MM_DD%	6	Format sous forme <Année sur 4 chiffres>, <Mois>, <Jour>

Ces valeurs sont employées par SETDATEFORMAT et retournées par GETDATEFORMAT%. Deux autres formats sont possibles :

- DD_MM_YY% ou DD_MM_YYYY%
- MM_DD_YY% ou MM_DD_YYYY%

Voir aussi DD MM YY%, MM DD YY*%*

Constantes HH_MM*% (Librairie NSDate)

Valeurs logiques employées pour définir le format d'heure.

Syntaxe	Déclaration interne	Description
HH_MM%	1	Heure exprimée en heure/minute sur 24 heures
HH_MM_AMPM%	2	Heure exprimée en heure/minute sur 12 heures avec symboles complémentaires pour indiquer avant ou après midi.
HH_MM_SS%	3	Heure exprimée en heure/minute/seconde sur 24 heures
HH_MM_SS_AMPM%	4	Heure exprimée en heure/minute/seconde sur 12 heures avec symboles complémentaires pour indiquer avant ou après midi.

1. Ces valeurs sont employées par SETTIMEFORMAT et retournées par GETTIMEFORMAT%.

2. Les symboles complémentaires d'un format de type HH_MM_AMPM% ou HH_MM_SS_AMPM% doivent être spécifiés avec SETTIMEAMPM s'ils ne sont pas définis dans le Panneau de Configuration.

Instruction SETDATEFORMAT (Librairie NSDate)

Précise le format employé pour les dates.

Syntaxe	SETDATEFORMAT <i>format</i>		
Paramètre	format	INT(1)	I format de date

1. Ce format est uniquement pris en compte au sein de l'application : cela ne modifie pas le format par défaut précisé dans le Panneau de Configuration et qui reste valable pour toutes les autres applications n'ayant pas spécifié de format particulier.

2. Les formats acceptés par SETDATEFORMAT sont :

- DD MM YY%,
- DD MM YYYY%,
- MM DD YY%,
- MM DD YYYY%,
- YY MM DD%,
- YYYY MM DD%.

3. SETDATEFORMAT 0 permet d'utiliser le format de date défini dans le Panneau de Configuration.

Exemple 1 :

```
; Supposons que la date est le 2 janvier 1995
SETDATEFORMAT YY_MM_DD%
SETDATESEPARATOR " / "
MOVE DATE$(CURRENTDATE%) TO A$ ; A$ vaut "95 / 1 / 2"
```

Exemple 2 :

```
; Reprise du format spécifié dans le Panneau de Configuration
SETDATEFORMAT 0
```

Voir aussi SETDATESEPARATOR, GETDATEFORMAT%, GETDATESEPARATOR\$, SETTIMEFORMAT, SETTIMESEPARATOR

Instruction SETDATEFORMAT2 (Librairie NSDate)

Précise le format employé pour les dates et le type d'affichage pour les jours et les mois.

Syntaxe	SETDATEFORMAT2 <i>format, day-leading-zero, month-leading-zero</i>		
Paramètres	format	INT(1)	I format de date

	num-jours-préfixé	INT(1)	I	indicateur de préfixage par zéro des numéros de jour.
	num-mois-préfixé	INT(1)	I	indicateur de préfixage par zéro des numéros de mois compris entre 1 et 9.

1. Le format de date spécifié par cette instruction est identique à celui spécifié à l'aide de l'instruction SETDATEFORMAT. Voir les commentaires de celle-ci.
2. Les valeurs de num-jours-préfixé et num-mois-préfixé (TRUE% ou FALSE%) permettent d'indiquer si les numéros de jour et de mois doivent être ou non précédés par un zéro lors de leur affichage.

Exemple :

```
; Supposons que la date courante soit le 7 Juin 1995
SETDATEFORMAT2 DD_MM_YYYY%,TRUE%,FALSE%
SETDATESEPARATOR "-"
MOVE DATE$(CURRENTDATE%) TO A$ ; A$ vaut "07-6-1995"
```

Voir aussi GETDATEFORMAT2%, SETDATESEPARATOR

Instruction SETDATESEPARATOR (Librairie NSDate)

Précise le séparateur employé pour les dates.

Syntaxe	SETDATESEPARATOR separator		
Paramètre	separator	CSTRING	I séparateur de date

1. Ce séparateur du format date est uniquement pris en compte au sein de l'application : cela ne modifie pas le séparateur par défaut précisé dans le

Panneau de Configuration et qui reste valable pour toutes les autres applications n'ayant pas spécifié de séparateur particulier.

- 2. SETDATESEPARATOR accepte toute chaîne de caractères comme séparateur.
- 3. SETDATESEPARATOR « ! » permet d'utiliser le séparateur de date défini dans le Panneau de Configuration.

Exemple 1 :

```
; Supposons que la date courante soit le 20 janvier 1995
SETDATEFORMAT YY_MM_DD%
SETDATESEPARATOR " / "
MOVE DATE$(CURRENTDATE%) TO A$ ; A$ vaut "95 / 1 / 20"
```

Exemple 2 :

```
; Reprise du format spécifié dans le Panneau de Configuration
SETDATESEPARATOR « ! »
```

Voir aussi SETDATEFORMAT, GETDATEFORMAT%, GETDATESEPARATOR\$, SETTIMEFORMAT, SETTIMESEPARATOR

Récupérer et paramétrer les détails d'un format

Fonction GETDATEFORMAT% (Librairie NSDate)

Retourne le format employé pour les dates au sein de l'application

Syntaxe	GETDATEFORMAT%
Valeur retournée	INT(1)

- 1. Le format retourné est égal à celui précisé dans le Panneau de Configuration tant qu'il n'a pas été modifié à l'aide de SETDATEFORMAT.
- 2. Les formats pouvant être retournés par GETDATEFORMAT% sont :
 - DD MM YY%,
 - DD MM YYYY%,
 - MM DD YY%,
 - MM DD YYYY%,
 - YY MM DD%,
 - YYYY MM DD%.

Exemple :

```
IF GETDATEFORMAT% = MM_DD_YYYY%
IF GETDATESEPARATOR$ = "/"
MESSAGE "Date", "Le format est MM/DD/YYYY"
ENDIF
ENDIF
```

Voir aussi [GETDATESEPARATOR\\$](#), [SETDATEFORMAT](#), [SETDATESEPARATOR](#), [GETTIMEFORMAT%](#), [GETTIMESEPARATOR\\$](#)

Fonction GETDATEFORMAT2% (Librairie NSDate)

Retourne le format employé pour les dates au sein de l'application ainsi que le format d'affichage des jours et des mois.

Syntaxe	GETDATEFORMAT2% (day-leading-zero, month-leading-zero)			
Paramètres	day-leading-zero	INT(1)	I/O	indicateur de préfixage par zéro des numéros de jour.
	month-leading-zero	INT(1)	I/O	indicateur de préfixage par zéro des numéros de mois compris entre 1 et 9.
Valeur retournée	INT(1)			

1. Le format de date retourné par cette fonction est identique à celui retourné par la fonction [GETDATEFORMAT%](#). Voir les commentaires de celle-ci.
2. Les valeurs de num-jours-préfixé et num-mois-préfixé (TRUE% ou FALSE%) permettent d'indiquer si les numéros de jour et de mois doivent être ou non précédés par un zéro lors de leur affichage.

Exemple :

```
LOCAL INT DLZ%(1) ; indicateur de préfixage pour les jours
LOCAL INT MLZ%(1) ; indicateur de préfixage pour les mois

IF GETDATEFORMAT2%(DLZ%,MLZ%) = MM_DD_YYYY%
IF DLZ% AND MLZ%
MESSAGE "Date", "Le format de date est en Jour,Mois,Année" & "avec un préfixage par zéro
pour les jours" && "et les mois"
ENDIF
ENDIF
```

Voir aussi [SETDATEFORMAT2](#), [SETDATESEPARATOR](#), [GETDATESEPARATOR\\$](#), [GETTIMEFORMAT%](#), [GETTIMESEPARATOR\\$](#)

Fonction GETDATESEPARATOR\$ (Librairie NSDate)

Retourne le séparateur employé pour les dates au sein de l'application.

Syntaxe	GETDATESEPARATOR\$
Valeur retournée	CSTRING

1. Le séparateur retourné est égal à celui précisé dans le Panneau de Configuration tant qu'il n'a pas été modifié à l'aide de SETDATESEPARATOR.
2. GETDATESEPARATOR\$ peut retourner toute chaîne de caractères comme séparateur.

Exemple :

```
IF GETDATEFORMAT% = MM_DD_YYYY%
IF GETDATESEPARATOR$ = "/"
MESSAGE "Date", "Le format est MM/DD/YYYY"
ENDIF
ENDIF
```

Voir aussi GETDATEFORMAT%, SETDATEFORMAT, SETDATESEPARATOR, GETTIMEFORMAT%, GETTIMESEPARATOR\$

Instruction SETTIMEFORMAT (Librairie NSDate)

Précise le format employé pour les heures.

Syntaxe	SETTIMEFORMAT <i>format</i>		
Paramètre	format	INT(1)	format d'heure

1. Ce format est uniquement pris en compte au sein de l'application : cela ne modifie pas celui pouvant être défini dans le Panneau de Configuration et qui reste valable pour toutes les autres applications n'ayant pas spécifié de format d'heure particulier.
2. Les formats acceptés par SETTIMEFORMAT sont :
 - HH MM%,
 - HH MM SS%,
 - HH MM AMPM%,
 - HH MM SS AMPM%.
3. Pour HH_MM% et HH_MM_SS%, les heures peuvent varier de 0 à 23.
4. Pour HH_AM_AMPM% et HH_MM_SS_AMPM%, les heures ne peuvent varier qu'entre 0 et 11 et les symboles précisés par SETTIMEAMPM sont rajoutés en fin de chaîne pour différencier le matin de l'après-midi.
5. SETTIMEFORMAT 0 permet d'utiliser le format d'heure défini dans le Panneau de Configuration

Exemple :

```
; Supposons que l'heure soit 20h 30mn 21s
SETTIMESEPARATOR " " : "
SETTIMEFORMAT HH_MM%
MOVE TIME$(CURRENTTIME) TO A$
; A$ vaut "20 : 30"
```

; Reprise du format spécifié dans le Panneau de Configuration
SETTIMEFORMAT 0

Voir aussi SETTIMEAMPM, SETTIMESEPARATOR, GETTIMEAM\$, GETTIMEPM\$, GETTIMEFORMAT%, GETTIMESEPARATOR\$, SETDATEFORMAT, SETDATESEPARATOR

Fonction GETTIMEFORMAT% (Librairie NSDate)

Retourne le format d'affichage des heures au sein de l'application.

Syntaxe	GETTIMEFORMAT%
Valeur retournée	INT(1)

1. La valeur retournée correspond au format précisé dans le Panneau de Configuration tant que SETTIMEFORMAT n'a pas été utilisé.
2. Les valeurs retournées par GETTIMEFORMAT% sont les constantes de format d'heure :
 - HH MM%,
 - HH MM AMPM%,
 - HH MM SS%,
 - HH MM SS AMPM%.

Exemple :

```
IF GETTIMEFORMAT% = HH_MM_SS%
IF GETTIMESEPARATOR$ = ":"
MESSAGE "Heure", "Le format est HH:MM:SS"
ENDIF
ENDIF
```

Voir aussi GETTIMEAM\$, GETTIMEPM\$, GETTIMESEPARATOR\$, SETTIMEAMPM, SETTIMEFORMAT, SETTIMESEPARATOR, GETDATEFORMAT%, GETDATESEPARATOR\$

Instruction SETTIMEAMPM (Librairie NSDate)

Précise les symboles utilisés pour indiquer une heure avant et après midi dans un format heure.

Syntaxe	SETTIMEAMPM am, pm		
Paramètres	am	CSTRING	symbole pour heure avant midi
	pm	CSTRING	symbole pour

			heure après midi
--	--	--	------------------------

1. Ces symboles sont uniquement pris en compte au sein de l'application : cela ne modifie pas ceux pouvant être définis dans le Panneau de Configuration et qui restent valable pour toutes les autres applications n'ayant pas spécifié de symboles particuliers.
2. SETTIMEAMPM accepte toutes chaînes de caractères.
3. Les symboles ainsi définis n'apparaissent que si le format de date est défini sur 12 heures dans le Panneau de Configuration ou si le format de date défini par SETTIMEFORMAT est de type HH MM AMPM% ou HH MM SS AMPM%.
4. SETTIMEAMPM "!", "!" permet d'utiliser les définitions du Panneau de Configuration.

Exemple :

```
SETTIMEAMPM "Matin", "Après-Midi"
SETTIMEFORMAT HH_MM_SS_AMPM%
MOVE TIME$(CURRENTTIME%) TO A$
; A$ vaut "08:30:21 Après-Midi"
; Reprise du format spécifié dans le panneau de configuration
SETTIMEAMPM "!", "!"
```

Voir aussi SETTIMEFORMAT, SETTIMESEPARATOR, GETTIMEAM\$, GETTIMEPM\$, GETTIMEFORMAT%, GETTIMESEPARATOR\$, SETDATEFORMAT, SETDATESEPARATOR

Fonction GETTIMEAM\$ (Librairie NSDate)

Retourne les symboles utilisés au sein de l'application pour identifier les heures avant midi lorsque celles-ci ne s'expriment que sur 12 heures.

Syntaxe	GETTIMEAM\$
Valeur retournée	CSTRING

1. La chaîne retournée reprend les définitions du Panneau de Configuration tant que SETTIMEAMPM n'a pas été utilisé.
2. Une chaîne vide est retournée lorsque le format d'affichage de l'heure dans le Panneau de Configuration n'est pas exprimé dans le format 12 heures ou que les symboles n'ont pas été définis par SETTIMEAMPM.

Exemple :

```
SETTIMEAMPM "AM", "PM"
MESSAGE GETTIMEAM$, GETTIMEPM$ ; affiche "AM" et "PM"
```

Voir aussi GETTIMEFORMAT%, GETTIMEPM\$, GETTIMESEPARATOR\$, SETTIMEAMPM, SETTIMEFORMAT, SETTIMESEPARATOR, GETDATEFORMAT%, GETDATESEPARATOR\$

Fonction GETTIMEPM\$ (Librairie NSDate)

Retourne les symboles utilisés au sein de l'application pour identifier les heures après midi lorsque celles-ci ne s'expriment que sur 12 heures.

Syntaxe	GETTIMEPM\$
Valeur retournée	CSTRING

1. La chaîne retournée reprend les définitions du Panneau de Configuration tant que SETTIMEAMPM n'a pas été utilisé.
2. Une chaîne vide est retournée lorsque le format d'affichage de l'heure dans le Panneau de Configuration n'est pas exprimée dans le format 12 heures ou que les symboles n'ont pas été définis par SETTIMEAMPM.

Exemple :

```
SETTIMEAMPM "AM", "PM"
MESSAGE GETTIMEAM$, GETTIMEPM$ ; displays "AM" and "PM"
```

Voir aussi GETTIMEAM\$, GETTIMEFORMAT%, GETTIMESEPARATOR\$, SETTIMEAMPM, SETTIMEFORMAT, SETTIMESEPARATOR, GETDATEFORMAT%, GETDATESEPARATOR\$

Instruction SETTIMESEPARATOR (Librairie NSDate)

Précise le séparateur employé pour les heures.

Syntaxe	SETTIMESEPARATOR <i>séparateur</i>		
Paramètre	séparateur	CSTRING	séparateur d'heure

1. Ce séparateur est uniquement pris en compte au sein de l'application : cela ne modifie pas celui pouvant être défini dans le Panneau de Configuration et qui reste valable pour toutes les autres applications n'ayant pas spécifié de séparateur de format d'heure particulier.
2. SETTIMESEPARATOR accepte toutes chaînes de caractères.
3. SETTIMESEPARATOR "!" permet d'utiliser le séparateur de format d'heure défini dans le Panneau de Configuration.

Exemple :

```
; Supposons que l'heure soit 20h 30mn 21s
SETTIMESEPARATOR " : "
SETTIMEFORMAT HH_MM%
MOVE TIME$(CURRENTTIME%) TO A$
; A$ vaut "20 : 30"
; Reprise du format spécifié dans le Panneau de Configuration
SETTIMESEPARATOR "!"
```

Voir aussi [SETTIMEAMPM](#), [SETTIMEFORMAT](#), [GETTIMEAM\\$](#), [GETTIMEPM\\$](#), [GETTIMEFORMAT%](#), [GETTIMESEPARATOR\\$](#), [SETDATEFORMAT](#), [SETDATESEPARATOR](#)

Fonction [GETTIMESEPARATOR\\$](#) (Librairie NSDate)

Retourne le séparateur employé dans le format des heures au sein de l'application.

Syntaxe	GETTIMESEPARATOR\$
Valeur retournée	CSTRING

La chaîne retournée correspond au séparateur précisé dans le Panneau de Configuration tant que [SETTIMESEPARATOR](#) n'a pas été utilisé.

Exemple :

```
IF GETTIMEFORMAT% = HH_MM_SS%
IF GETTIMESEPARATOR$ = ":"
MESSAGE "Heure", "Le format est HH:MM:SS"
ENDIF
ENDIF
```

Voir aussi [GETTIMEAM\\$](#), [GETTIMEFORMAT%](#), [GETTIMEPM\\$](#), [SETTIMEAMPM](#), [SETTIMEFORMAT](#), [SETTIMESEPARATOR](#), [GETDATEFORMAT%](#), [GETDATESEPARATOR\\$](#)

Récupérer et paramétrer les noms de jour et de mois

Instruction [SETLONGDAY](#) (Librairie NSDate)

Donne un nom à un jour de la semaine.

Syntaxe	SETLONGDAY entier-jour, chaîne-jour		
Paramètres	entier-jour	INT(1)	jour de la semaine
	chaîne-jour	CSTRING	nom du jour

- entier-jour doit être compris entre 0 (correspondant au Dimanche) et à 6 (correspondant au Samedi).
- Le nom du jour ainsi défini celui qui est ensuite retourné par [GETLONGDAY\\$](#).
- Cette fonction permet de spécifier le nom de chaque jour de la semaine dans une autre langue que l'anglais, langue par défaut de [GETLONGDAY\\$](#).

Exemple :


```
; Les jours en français sont :
SETLONGDAY 0, "Dimanche"
SETLONGDAY 1, "Lundi"
SETLONGDAY 2, "Mardi"
SETLONGDAY 3, "Mercredi"
SETLONGDAY 4, "Jeudi"
SETLONGDAY 5, "Vendredi"
SETLONGDAY 6, "Samedi"
MESSAGE "Test jour", GETLONGDAY$(3) ; affiche Mercredi
```

Voir aussi [SETLONGMONTH](#), [GETLONGDAY\\$](#), [GETLONGMONTH\\$](#), [SETSHORTDAY](#), [SETSHORTMONTH](#)

Fonction GETLONGDAY\$ (Librairie NSDate)

Retourne le nom du jour de la semaine correspondant à la valeur passée en paramètre.

Syntaxe	GETLONGDAY\$ (day)		
Paramètre	day	INT(1)	I entier correspondant au jour de la semaine
Valeur retournée	CSTRING		

1. Lorsque jour est en dehors des limites autorisées (0, pour Dimanche, à 6, pour Samedi), cette fonction retourne une chaîne vide.
2. Par défaut, la chaîne retournée donne le nom du jour en langue anglaise :

Jour	Chaîne retournée
0	"Sunday"
1	"Monday"
2	"Tuesday"
3	"Wednesday"
4	"Thursday"
5	"Friday"
6	"Saturday"
>=7	""

3. Utilisez au préalable l'instruction [SETLONGDAY](#) pour définir le nom des jours de la semaine dans une autre langue.

Exemple :

```
MESSAGE "Le jour d'aujourd'hui est :", GETLONGDAY$(CURRENTDAY%)
```

Voir aussi [GETLONGMONTH\\$](#), [SETLONGDAY](#), [SETLONGMONTH](#), [GETSHORTDAY\\$](#), [GETSHORTMONTH\\$](#)

Instruction SETLONGMONTH (Librairie NSDate)

Donne un nom à chaque mois de l'année.

Syntaxe	SETLONGMONTH <i>entier-mois, chaîne-mois</i>			
Paramètres	entier-mois	INT(1)	I	mois
	chaîne-mois	CSTRING	I	nom du mois

1. entier-mois doit être compris entre 0 (correspondant à Janvier) et à 11 (correspondant à Décembre).
2. Le nom du mois ainsi défini est celui qui est ensuite retourné par GETLONGMONTH\$
3. Cette fonction permet de spécifier le nom de chaque mois de l'année dans une autre langue que l'anglais, langue par défaut de GETLONGMONTH\$.

Exemple :

```
; Les mois en français sont :
SETLONGMONTH 0, "Janvier"
SETLONGMONTH 1, "Février"
SETLONGMONTH 2, "Mars"
SETLONGMONTH 3, "Avril"
SETLONGMONTH 4, "Mai"
SETLONGMONTH 5, "Juin"
SETLONGMONTH 6, "Juillet"
SETLONGMONTH 7, "Août"
SETLONGMONTH 8, "Septembre"
SETLONGMONTH 9, "Octobre"
SETLONGMONTH 10, "Novembre"
SETLONGMONTH 11, "Décembre"
MESSAGE "Mois #5 est: ", GETLONGMONTH$(5) ; affiche Juin
```

Voir aussi SETLONGDAY, GETLONGDAY\$, GETLONGMONTH\$, SETSHORTDAY, SETSHORTMONTH

Fonction GETLONGMONTH\$ (Librairie NSDate)

Retourne le nom du mois correspondant à la valeur passée en paramètre.

Syntaxe	GETLONGMONTH\$ (<i>month</i>)			
Paramètre	month	INT(1)	I	entier correspondant au mois
Valeur retournée	CSTRING			

1. Lorsque mois est en dehors des limites autorisées (0, pour Janvier, à 11, pour Décembre), cette fonction retourne une chaîne vide.
2. Par défaut, la chaîne retournée donne le nom du mois en langue anglaise

Mois	Chaîne retournée
------	------------------

0	"January"
1	"February"
2	"March"
3	"April"
4	"May"
5	"June"
6	"July"
7	"August"
8	"September"
9	"October"
10	"November"
11	"December"
>=12	""

3. Utilisez au préalable l'instruction SETLONGMONTH pour définir le nom des mois dans une autre langue.

Exemple :

```
MOVE "/" TO SEP$
SETDATEFORMAT MM_DD_YY%
SETDATESEPARATOR SEP$
MOVE DATE$(CURRENTDATE%) TO TODAY$
MESSAGE "Nous sommes le mois de :", GETLONGMONTH$(INT COPY$(TODAY$,1,POS$(SEP$,TODAY$)-1)-1)
```

Voir aussi GETLONGDAY\$, SETLONGDAY, SETLONGMONTH, GETSHORTDAY\$, GETSHORTMONTH\$

Instruction SETSHORTDAY (Librairie NSDate)

Donne une abréviation au nom d'un jour de la semaine.

Syntaxe	SETSHORTDAY entier-jour, chaîne-jour		
Paramètres	entier-jour	INT(1)	jour de la semaine
	chaîne-jour	CSTRING	abréviation du nom du jour

1. entier-jour doit être compris entre 0 (correspondant au Dimanche) et 6 (correspondant au Samedi).

2. L'abréviation du nom du jour ainsi définie est celle qui est ensuite retournée par `GETSHORTDAY$`. Elle n'est composée au maximum que des cinq premiers caractères de chaîne-jour.
3. Cette fonction permet de modifier les abréviations par défaut retournées par `GETSHORTDAY$` ou de spécifier une abréviation dans une langue différente de l'anglais.

Exemple :

```
; Les abréviations des jours en français sont :
SETSHORTDAY 0, "Dim"
SETSHORTDAY 1, "Lun"
SETSHORTDAY 2, "Mar"
SETSHORTDAY 3, "Mer"
SETSHORTDAY 4, "Jeu"
SETSHORTDAY 5, "Ven"
SETSHORTDAY 6, "Sam"
MESSAGE "Test jour", GETSHORTDAY$(3) ; affiche Mer
```

Voir aussi `SETSHORTMONTH`, `GETSHORTDAY$`, `GETSHORTMONTH$`, `SETLONGDAY`, `SETLONGMONTH`

Fonction `GETSHORTDAY$` (Librairie `NSDate`)

Retourne les trois premières lettres du jour de la semaine correspondant à la valeur passée en paramètre.

Syntaxe	<code>GETSHORTDAY\$(day)</code>
Paramètre	day INT(1) entier correspondant au jour de la semaine
Valeur retournée	CSTRING

1. Lorsque jour est en dehors des limites autorisées (0, pour Dimanche, à 6, pour Samedi), cette fonction retourne une chaîne vide.
2. Par défaut, la chaîne retournée donne les premières lettres du jour exprimé en langue anglaise :

Jour	Chaîne retournée
0	"Sun"
1	"Mon"
2	"Tue"
3	"Wed"
4	"Thu"
5	"Fri"
6	"Sat"

>=7	""
-----	----

3. Utilisez au préalable l'instruction SETSHORTDAY pour définir le nom des jours de la semaine dans une autre langue ou pour modifier une ou plusieurs chaînes retournées.

Exemple :

```
MESSAGE "Le jour d'aujourd'hui est :", GETSHORTDAY$(CURRENTDAY%)
```

Voir aussi GETSHORTMONTH\$, SETSHORTDAY, SETSHORTMONTH, GETLONGDAY\$, GETLONGMONTH\$

Instruction SETSHORTMONTH (Librairie NSDate)

Donne une abréviation du nom d'un mois.

Syntaxe	SETSHORTMONTH entier-mois, chaîne-mois		
Paramètres	entier-mois	INT(1)	mois
	chaîne-mois	CSTRING	abréviation du mois

- entier-mois doit être compris entre 0 (correspondant à Janvier) et 11 (correspondant à Décembre).
- L'abréviation du nom du mois ainsi définie est celle qui est ensuite retournée par GETSHORTMONTH\$. Elle n'est composée au maximum que des cinq premiers caractères de chaîne-mois.
- Cette fonction permet de modifier les abréviations par défaut retournées par GETSHORTMONTH\$ ou de spécifier une abréviation dans une langue différente de l'anglais.

Exemple :

```
; Les abréviations des mois en français sont :
SETSHORTMONTH 0, "Jan"
SETSHORTMONTH 1, "Fév"
SETSHORTMONTH 2, "Mar"
SETSHORTMONTH 3, "Avr"
SETSHORTMONTH 4, "Mai"
SETSHORTMONTH 5, "Jun"
SETSHORTMONTH 6, "Juil"
SETSHORTMONTH 7, "Août"
SETSHORTMONTH 8, "Sep"
SETSHORTMONTH 9, "Oct"
SETSHORTMONTH 10, "Nov"
SETSHORTMONTH 11, "Déc"
MESSAGE "Test mois", GETSHORTMONTH$(5) ; affiche Jun
```

Voir aussi SETSHORTDAY, GETSHORTDAY\$, GETSHORTMONTH\$, SETLONGDAY, SETLONGMONTH

Fonction GETSHORTMONTH\$ (Librairie NSDate)

Retourne les trois premières lettres du mois correspondant à la valeur passée en paramètre.

Syntaxe	GETSHORTMONTH\$ (<i>month</i>)		
Paramètre	month	INT(1)	I entier correspondant au mois
Valeur retournée	CSTRING		

1. Lorsque mois est en dehors des limites autorisées (0, pour Janvier, à 11, pour Décembre), cette fonction retourne une chaîne vide.
2. Par défaut, la chaîne retournée donne les premières lettres du mois exprimé en langue anglaise :

Mois	Chaîne retournée
0	"Jan"
1	"Feb"
2	"Mar"
3	"Apr"
4	"May"
5	"Jun"
6	"Jul"
7	"Aug"
8	"Sep"
9	"Oct"
10	"Nov"
11	"Dec"
>=12	""

3. Utilisez au préalable l'instruction SETSHORTMONTH pour définir le nom des mois dans une autre langue ou pour modifier une ou plusieurs chaînes retournées.

Exemple :

```
MOVE "/" TO SEP$
SETDATEFORMAT MM_DD_YY%
SETDATESEPARATOR SEP$
MOVE DATE$(CURRENTDATE%) TO TODAY$
MESSAGE "Nous sommes le mois de :", GETSHORTMONTH$(INT COPY$(TODAY$,1,POS%(SEP$,TODAY$) - 1)-1)
```

Voir aussi [GETSHORTDAY\\$](#), [SETSHORTDAY](#), [SETSHORTMONTH](#), [GETLONGDAY\\$](#), [GETLONGMONTH\\$](#)

Vérifier un format

Fonction ISDATE% ([Librairie NSDate](#))

Teste si une chaîne correspond au format date courant.

Syntaxe	ISDATE% (date)			
Paramètre	date	CSTRING	I	chaîne à tester
Valeur retournée	INT(1) TRUE% : la chaîne est une date au bon format FALSE% : la chaîne n'est pas au bon format			

Le format date de référence est celui spécifié par [SETDATEFORMAT](#) / [SETDATESEPARATOR](#) ou, par défaut, celui défini dans le Panneau de Configuration.

Voir aussi [ISTIME%](#)

Fonction ISTIME% ([Librairie NSDate](#))

Teste si une chaîne correspond au format heure courant.

Syntaxe	ISTIME% (time)			
Paramètre	time	CSTRING	I	chaîne à tester
Valeur retournée	INT(1) TRUE% : la chaîne est une heure au bon format FALSE% : la chaîne n'est pas au bon format			

Le format heure de référence est celui spécifié par [SETDATEFORMAT](#) / [SETDATESEPARATOR](#) ou, par défaut, celui défini dans le Panneau de Configuration.

Voir aussi [ISDATE%](#)

Convertir de et vers des entiers et chaînes de caractères

Fonction DATE\$ ([Librairie NSDate](#))

Convertir un entier en une date.

Syntaxe	DATE\$ (number)			
Paramètre	number	INT(4)	I	entier à convertir en date
Valeur retournée	CSTRING			

1. number contient le nombre de jours écoulés depuis le 01/01/1900.
2. La date est renvoyée dans le format de date courant. Ce format est celui spécifié par SETDATEFORMAT et SETDATESEPARATOR ou, par défaut, celui défini dans le Panneau de Configuration.
3. La fonction retourne "#ERR!" si la valeur passée est incorrecte.
4. La fonction STRING\$ permet aussi de formater un entier en une date. Voir l'exemple.
5. La librairie NSDate sait gérer toutes les dates à partir du 1/1/1583 (début du calendrier grégorien) en autorisant des entiers négatifs pour les dates antérieures au 1/1/1900. Ainsi :

```

DATE$(-115784) retourne "#ERR!"
DATE$(-115783) retourne "#ERR!"
DATE$(-115782) retourne "1/1/1583"
DATE$(-115781) retourne "2/1/1583"
...
DATE$(-1) retourne "31/12/1899"
DATE$(0) retourne "1/1/1900"
DATE$(1) retourne "2/1/1900"
DATE$(2) retourne "3/1/1900"
...

```

Exemple 1 :

```

S$ = DATE$(CURRENTDATE% + 1); S$ vaut la date de demain

```

Exemple 2 :

```

; définition du format de date format de date sous forme année mois jour
SETDATEFORMAT YYYY_MM_DD%
; le séparateur utilisé est le tiret
SETDATESEPARATOR "-"

S$ = DATE$(0); S$ vaut "1900-1-1"
S$ = MOVE DATE$(33602); S$ vaut "1992-1-1"
Exemple 3
; STRING$ de la librairie NSDate peut aussi être employé
S$ = STRING$(33602, "yyyy-m-d"); S$ vaut "1992-1-1"

```

Voir aussi DATE%, DATE_ERROR%, CURRENTDATE%, TIME\$, TIME%, STRING\$ (Librairie NSMisc)

Fonction DATE% (Librairie NSDate)

Convertit une date en entier.

Syntaxe	DATE% (date)			
Paramètre	date	CSTRING	I	date à convertir en entier
Valeur retournée	INT(4) contient le nombre de jours écoulés depuis le 01/01/1900.			

1. La date doit être passée dans le format de date courant. Ce format est celui spécifié par SETDATEFORMAT et SETDATESEPARATOR ou, par défaut, celui défini dans le panneau de Configuration.
2. La fonction retourne DATE_ERROR% si la date passée est dans un format incorrect.
3. La librairie NSDate sait gérer toutes les dates à partir de 1/1/1583 (début du calendrier grégorien) en autorisant des entiers négatifs pour les dates antérieures au 1/1/1900.

Exemple :

```
; format de date sous forme année mois jour
SETDATEFORMAT YYYY_MM_DD%
; le séparateur utilisé est le tiret
SETDATESEPARATOR « - »

I% = DATE%(« 1990-1-1 ») ; I% vaut 32872
I% = DATE%(« 1991-1-1 ») ; I% vaut 33237
```

Voir aussi DATE\$, DATE_ERROR%, CURRENTDATE%, TIME\$, TIME%

Fonction **TIME\$** (Librairie NSDate)

Convertit un entier en une heure.

Syntaxe	TIME\$ (nombre)			
Paramètre	nombre	INT(4)	I	entier à convertir en heure
Valeur retournée	CSTRING			

1. nombre doit contenir le nombre de secondes écoulées depuis 0h 0mn 0s.
2. L'heure est renvoyée dans le format d'heure courant. Ce format est celui spécifié par SETTIMEFORMAT et SETTIMESEPARATOR ou, par défaut, celui défini dans le Panneau de Configuration.
3. La fonction retourne "#ERR!" si la valeur passée est incorrecte.
4. La fonction STRING\$ de la librairie NSMisc permet aussi de formater un entier en une heure. Voir l'exemple.
5. La librairie NSDate sait gérer toutes les heures entre 00:00:00 et 23:59:59. TIME\$ retourne "#ERR!" dès que la valeur passée est négative (< 00:00:00) ou supérieure à 86399 (> 23:59:59).

Exemple 1 :

```
SETTIMEFORMAT HH_MM_SS%
SETTIMESEPARATOR ":"
S$ = TIME$(0); S$ vaut "00:00:00"
S$ = TIME$(43200) ; S$ vaut "12:00:00"
```

Exemple 2 :

```
; STRING$ de la librairie NSDATE peut aussi être employé  
S$ = STRING$(43200, "hh:mm:ss") ; S$ vaut "12:00:00"
```

Voir aussi TIME%, TIME_ERROR%, CURRENTTIME%, DATE\$, DATE%, STRING\$ (librairie NSMisc)

Fonction TIME% (Librairie NSDate)

Convertit une heure en un entier.

Syntaxe	TIME% (heure)			
Paramètre	heure	CSTRING	I	heure à convertir en entier
Valeur retournée	INT(4)			

1. L'heure est envoyée dans le format d'heure courant. Ce format est celui spécifié par SETTIMEFORMAT et SETTIMESEPARATOR ou, par défaut, celui défini dans le Panneau de Configuration.
2. La fonction STRING\$ de la librairie NSMisc permet aussi de formater un entier en une heure. Voir l'exemple.
3. La librairie NSDate sait gérer toutes les heures entre 00:00:00 et 23:59:59. TIME\$ retourne "#ERR!" dès que la valeur passée est négative (< 00:00:00) ou supérieure à 86399 (> 23:59:59).

Exemple :

```
SETTIMEFORMAT HH_MM_SS%  
SETTIMESEPARATOR ":"  
MOVE TIME%("09:00:00") TO I% ; I% vaut 32400  
MOVE TIME%("23:59:59") TO I% ; I% vaut 86399 (valeur maximum que puisse retourner TIME%)
```

Voir aussi TIME\$, TIME_ERROR%, CURRENTTIME%, DATE\$, DATE%, STRING\$ (librairie NSMisc)

Année jalon

Instruction SETMILESTONEYEAR (Librairie NSDate)

Fixe l'année jalon permettant de faire basculer une date entre les années 1900 et les années 2000.

Dans les versions antérieures à NatStar 2.16, une date spécifiée avec une année à deux chiffres (01/01/13) était considéré comme une date du siècle courant (01/01/1913). L'année jalon permet de basculer dans un autre siècle.

Pour toute date à deux chiffres :

- Si l'an est plus petit à l'an jalon, alors l'année est après l'an 2000
- Sinon, l'année est considérée comme située avant l'an 2000.

Par exemple, pour un an jalon égal à 30 :

- La saisie d'une date 010129 donnera en sortie 01/01/2029
- La saisie de 010130 donnera en sortie 01/01/1930
- La saisie de 010131 donnera en sortie 01/01/1931

Syntaxe	SETMILESTONEYEAR <i>milestoneyear%</i>		
Paramètre	milestoneyear%	INT(4)	Valeur de l'année jalon (entier compris entre 0 et 99)

- L'année jalon peut également être définie dans le fichier NSLIB.INI à la section [Year2000]

```
[Year2000]
;Force4Digits=False
MilestoneYear=30
```

- La valeur de l'an jalon est lue à l'initialisation. Si aucune valeur n'est spécifiée dans le fichier, la valeur par défaut sera -1. La valeur par défaut correspond à l'ancien comportement, c'est-à-dire que pour compléter une année à 2 chiffres, on prendra le siècle courant.

Voir aussi [GETMILESTONEYEAR%](#)

Fonction GETMILESTONEYEAR% (Librairie NSDate)

Retourne la valeur de l'année jalon.

Syntaxe	GETMILESTONEYEAR%
Valeur retournée	INT(4)

Voir aussi [SETMILESTONEYEAR](#)

Gestion d'erreurs (constantes retournées)

Constante DATE_ERROR% (Librairie NSDate)

Syntaxe	Déclaration interne	Description
---------	---------------------	-------------

DATE_ERROR%	-115783	Code d'erreur retourné en cas de date incorrecte.
-------------	---------	---

Cette valeur est retournée par la fonction DATE% si la chaîne qui lui est passée en paramètre ne correspond pas à une date.

Voir aussi DATE%, TIME_ERROR%

Constantes TIME_ERROR% (Librairie NSDate)

Syntaxe	Déclaration interne	Description
TIME_ERROR%	-1	Code d'erreur retourné en cas d'heure incorrecte.

Cette valeur est retournée par la fonction TIME% si la chaîne qui lui est passée en paramètre ne correspond pas à une heure.

Voir aussi TIME%, DATE_ERROR%

LIBRAIRIE MATHÉMATIQUE NSMATH

Cette librairie permet de faire des calculs mathématiques (trigonométriques, logarithmiques...)

Installation

Déclarez NSMATH.NCL dans les librairies nécessaires au développement de votre application.

Vérifiez que le fichier NSxxMATH.DLL est bien dans un des répertoires du PATH sous Windows.

Extensions du langage NCL pour les calculs mathématiques

Les calculs trigonométriques s'obtiennent avec SIN#, COS#, ARCTAN#.

Les calculs logarithmiques s'obtiennent avec LN# et EXP#.

SQR# retourne le carré, alors que SQRT# retourne la racine carrée.

POWER# retourne un nombre X à la puissance Y.

FRAC# retourne la partie décimale, alors que INT# retourne la partie entière.

Et PI# retourne la valeur du nombre réel Pi.

L'ensemble des fonctions précédentes travaillent sur des entiers de type NUM(8). Elles ont chacune une fonction similaire travaillant sur des réels "extended" de type NUM(10). Chacune de ces fonctions a un nom préfixé par E. (ESIN#, ELN#, EFRAC#, etc).

Fichier NSMATH.NCL

Les verbes de la librairie NSMath, décrits ci après, sont déclarés dans le fichier texte écrit en NCL, de nom NSMATH.NCL. Ce fichier peut aussi contenir des verbes complémentaires (API non publique). Vous pouvez donc désirer consulter ce fichier pour avoir la référence exhaustive de la librairie.

Pour consulter le fichier NSMATH.NCL :

1. Placez vous dans le répertoire <NATSTAR>\NCL ou <NSDK>\NCL
<NATSTAR> représente le répertoire que vous avez choisi au moment de l'installation de NatStar et <NSDK> celui de NS-DK.
2. Editez le fichier NSMATH.NCL avec n'importe quel éditeur de texte.

Catégories fonctionnelles de la librairie NSMath

Partie entière et décimale d'un nombre réel

Fonction **FRAC#** (Librairie NSMath)

Retourne la partie décimale d'un nombre réel.

Syntaxe	FRAC# (x)			
Paramètre	x	NUM(8)		nombre réel
Valeur retournée	NUM(8)			

Utiliser EFRAC# si x est de type NUM(10).

Exemple :

```
LOCAL N#  
MOVE FRAC#(PI#) TO N# ; N# vaut 0.14159....
```

Voir aussi EFRAC#, INT#

Fonction **EFRAC#** (Librairie NSMath)

Retourne la partie décimale d'un nombre réel.

Syntaxe	EFRAC# (x)			
Paramètre	x	NUM(10)		nombre réel
Valeur retournée	NUM(10)			

Fonction identique à FRAC# mais manipulant des réels de type NUM(10).

Voir aussi FRAC#, EINT#

Fonction **INT#** (Librairie NSMath)

Retourne la partie entière d'un nombre réel.

Syntaxe	INT# (x)			
Paramètre	x	NUM(8)		nombre réel
Valeur retournée	NUM(8)			

Utiliser EINT# si x est de type NUM(10).

Exemple :

```
LOCAL N#  
MOVE INT#(PI#) TO N# ; N# vaut 3
```

Voir aussi EINT#, FRAC#

Fonction EINT# (Librairie NSMath)

Retourne la partie entière d'un nombre réel.

Syntaxe	EINT# (x)			
Paramètre	x	NUM(10)		nombre réel
Valeur retournée	NUM(10)			

Fonction identique à INT# mais manipulant des réels de type NUM(10).

Voir aussi INT#, EFAC#

Carré, racine carrée et puissances

Fonction POWER# (Librairie NSMath)

Retourne la valeur d'un réel X à la puissance Y.

Syntaxe	POWER# (x,y)			
Paramètres	x	NUM(8)		nombre réel
	y	NUM(8)		puissance
Valeur retournée	NUM(8)			

Utiliser EPOWER# si x et y sont de type NUM(10).

Exemple :

```
LOCAL N#
; calcul de 23
MOVE POWER#(2,3) TO N# ; N# vaut 8
```

Voir aussi EPOWER#

Fonction EPOWER# (Librairie NSMath)

Retourne la valeur d'un réel X à la puissance Y.

Syntaxe	EPOWER# (x, y)			
Paramètres	x	NUM(10)		nombre réel
	y	NUM(10)		puissance
Valeur retournée	NUM(10)			

Fonction identique à POWER# mais manipulant des réels de type NUM(10).

Voir aussi POWER#

Fonction SQR# (Librairie NSMath)

Retourne le carré d'un nombre réel.

Syntaxe	SQR# (x)			
Paramètre	x	NUM(8)		nombre réel
Valeur retournée	NUM(8)			

1. Ne pas confondre avec SQRT# (racine carrée)
2. Utiliser ESQR# si x est de type NUM(10).

Exemple :

```
LOCAL N#  
MOVE SQR#(3) TO N# ; N# vaut 9
```

Voir aussi ESQR#, SQRT#

Fonction ESQR# (Librairie NSMath)

Retourne le carré d'un nombre réel..

Syntaxe	ESQR# (x)			
Paramètre	x	NUM(10)		nombre réel
Valeur retournée	NUM(10)			

Fonction identique à SQR# mais manipulant des réels de type NUM(10).

Voir aussi ESQRT#, SQR#

Fonction SQRT# (Librairie NSMath)

Retourne la racine carrée d'un nombre réel (positif).

Syntaxe	SQRT# (x)			
Paramètre	x	NUM(8)		nombre réel
Valeur retournée	NUM(8)			

1. Ne pas confondre avec SQR# (carré)
2. Utiliser ESQRT# si x est de type NUM(10).

Exemple :


```
LOCAL N#
MOVE SQRT#(-1) TO N# ; Erreur !
MOVE SQRT#(9) TO N# ; N# vaut 3
```

Voir aussi ESQR#, SQR#

Fonction ESQR# (Librairie NSMath)

Retourne la racine carrée d'un nombre réel (positif).

Syntaxe	ESQR# (x)			
Paramètre	x	NUM(10)		nombre réel
Valeur retournée	NUM(10)			

Fonction identique à SQR# mais manipulant des réels de type NUM(10).

Voir aussi ESQR#, SQR#

Constante "Pi"

Fonction PI# (Librairie NSMath)

Retourne la valeur de Pi, soit 3.14159...

Syntaxe	PI# (x)			
Paramètre	x	NUM(8)		nombre réel
Valeur retournée	NUM(8)			

Utiliser EPI# si x est de type NUM(10).

Exemple :

```
LOCAL N#
; Calcul de la surface d'un cercle de rayon 2
MOVE PI# * SQR#(2) TO N# ; N# vaut 12.56637
```

Voir aussi EPI#

Fonction EPI# (Librairie NSMath)

Retourne la valeur de Pi, soit 3.14159...

Syntaxe	EPI#			
Valeur retournée	NUM(10)			

Fonction identique à PI# mais manipulant des réels de type NUM(10).

Voir aussi PI#

Fonctions trigonométriques

Fonction ARCTAN# (Librairie NSMath)

Retourne l'arc tangente d'un nombre réel.

Syntaxe	ARCTAN# (x)			
Paramètre	x	NUM(8)	I	nombre réel en radians
Valeur retournée	NUM(8)			

Utiliser EARCTAN# si x est de type NUM(10).

Exemple :

```
LOCAL N#  
MOVE ARCTAN# (1) TO N# ; N# vaut Pi/4, i.e. 0.78539...
```

Voir aussi EARCTAN#, COS#, SIN#

Fonction EARCTAN# (Librairie NSMath)

Retourne l'arc tangente d'un nombre réel.

Syntaxe	EARCTAN# (x)			
Paramètre	x	NUM(10)	I	nombre réel en radians
Valeur retournée	NUM(10)			

Fonction identique à ARCTAN# mais manipulant des réels de type NUM(10).

Voir aussi ARCTAN#, ECOS#, ESIN#

Fonction COS# (Librairie NSMath)

Retourne le cosinus d'un nombre réel.

Syntaxe	COS# (x)			
Paramètre	x	NUM(8)	I	nombre réel en radians
Valeur retournée	NUM(8)			

Utiliser ECOS# si x est de type NUM(10).

Exemple :

LOCAL N#

```
MOVE COS#(0) TO N# ; N# vaut 1
MOVE COS#(PI#/4) TO N# ; N# vaut (racine carrée de 2) / 2,
; soit 0.70710...
MOVE COS#(1) TO N# ; N# vaut 0.54030...
MOVE COS#(PI#/2) TO N# ; N# vaut 0
```

Voir aussi [ARCTAN#](#), [SIN#](#), [COS#](#)

Fonction ECOS# (Librairie NSMath)

Retourne le cosinus d'un nombre réel.

Syntaxe	ECOS# (x)			
Paramètre	x	NUM(10)		nombre réel en radians**
Valeur retournée	NUM(10)			

Fonction identique à [COS#](#) mais manipulant des réels de type NUM(10).

Voir aussi [COS#](#), [EARCTAN#](#), [ESIN#](#)

Fonction SIN# (Librairie NSMath)

Retourne le sinus d'un nombre réel.

Syntaxe	SIN# (x)			
Paramètre	x	NUM(8)		nombre réel en radian
Valeur retournée	NUM(8)			

Utiliser [ESIN#](#) si x est de type NUM(10).

Exemple :

LOCAL N#

```
MOVE SIN#(0) TO N# ; N# vaut 0
MOVE SIN#(PI#/4) TO N# ; N# vaut (racine carrée de 2) / 2,
; soit 0.70710...
MOVE SIN#(1) TO N# ; N# vaut 0.84147...
MOVE SIN#(PI#/2) TO N# ; N# vaut 1
```

Voir aussi [ARCTAN#](#), [COS#](#), [ESIN#](#)

Fonction ESIN# (Librairie NSMath)

Retourne le sinus d'un nombre réel.

Syntaxe	ESIN# (x)			
---------	-----------	--	--	--

Paramètre	x	NUM(10)		nombre réel en radian
Valeur retournée	NUM(10)			

Fonction identique à SIN# mais manipulant des réels de type NUM(10).

Voir aussi EARCTAN#, ECOS#, SIN#

Logarithmes et exponentielles

Fonction EXP# (Librairie NSMath)

Retourne l'exponentielle d'un nombre réel.

Syntaxe	EXP# (x)			
Paramètre	x	NUM(8)		nombre réel
Valeur retournée	NUM(8)			

Utiliser EEXP# si x est de type NUM(10).

Exemple :

```
LOCAL N#
MOVE EXP#(0) TO N# ; N# vaut 1
MOVE EXP#(1) TO N# ; N# vaut 2.71828...
MOVE EXP#(2) TO N# ; N# vaut 7.38905..
; (carré du nombre précédent)

; calcul de 23
MOVE EXP#(3*LN#(2)) TO N# ; N# vaut 8
; préférer POWER#(2,3)
```

Voir aussi EEXP#, LN#

Fonction EEXP# (Librairie NSMath)

Retourne l'exponentiel d'un nombre réel.

Syntaxe	EEXP# (x)			
Paramètre	x	NUM(10)		nombre réel
Valeur retournée	NUM(10)			

Fonction identique à EXP# mais manipulant des réels de type NUM(10).

Voir aussi EXP#, ELN#

Fonction LN# (Librairie NSMath)

Retourne le logarithme népérien d'un nombre réel (strictement positif).

Syntaxe	LN# (x)		
Paramètre	x	NUM(8)	I nombre réel
Valeur retournée	NUM(8)		

1. La fonction retourne 0 (zéro) si x est négatif ou nul.
2. Utiliser ELN# si x est de type NUM(10).

Exemple :

```

LOCAL N#
MOVE LN#(0) TO N# ; Erreur !!!
MOVE LN#(1) TO N# ; N# vaut 0
MOVE LN#(2) TO N# ; N# vaut 0.69314

; calcul de 23
MOVE EXP#(3*LN#(2)) TO N# ; N# vaut 8
; préférer POWER#(2,3)

```

Voir aussi ELN#, EXP#

Fonction ELN# (Librairie NSMath)

Retourne le logarithme népérien d'un nombre réel (strictement positif).

Syntaxe	ELN# (x)		
Paramètre	x	NUM(10)	I nombre réel
Valeur retournée	NUM(10)		

1. La fonction retourne 0 (zéro) si x est négatif ou nul.
2. Fonction identique à LN# mais manipulant des réels de type NUM(10).

Voir aussi LN#, EEXP#

LIBRAIRIE NSFEXE

Cette librairie permet de lancer des exécutables ou de démarrer des sessions "ligne de commande" depuis votre application. Les lancements peuvent être effectués en synchrone ou asynchrone avec, si nécessaire, récupération du code retour des exécutables soit dans une variable soit par notification envoyée à l'aide d'un message utilisateur dans une fenêtre de l'application.

Dans ce qui suit, le terme "console" désigne une session "ligne de commande" permettant de taper une commande système ou de démarrer un fichier batch. Cette session apparaît dans une fenêtre. Une telle session est souvent appelée "boîte DOS" sous Windows 3.x.

Installation

Déclarez NSFEXE.NCL dans les librairies nécessaires au développement de votre application.

Vérifiez que le fichier NSxxFEXE.DLL est bien dans un des répertoires du PATH sous Windows.

Extensions du langage NCL pour l'exécution de programmes

La librairie NSFexe comprend les verbes et constantes suivantes :

OPENCONSOLE% ouvre une console DOS dans laquelle pourront s'exécuter des programmes et dont les notifications sont envoyées à une fenêtre NatStar (ou NS-DK). CLOSECONSOLE% ferme cette console.

STARTCONSOLEEXECEX% démarre un programme dans une console DOS.

STOPCONSOLEEXEC% arrête le programme démarré dans cette console.

STARTEXCUTEEX% démarre un programme dont la sortie peut être redirigée vers une fenêtre NatStar (ou NS-DK). KILLEXCUTE et STOPEXCUTEEX arrêtent le programme démarré par cette fonction.

STARTEXCUTE2% démarre un programme en retournant l'identifiant de la session démarrée. STOPEXCUTE2 arrête le programme démarré par cette fonction.

Les constantes CON NFY *% et EXEC NFY TERMINATE% sont utilisées lors des notifications envoyées à la fenêtre NatStar (ou NS-DK).

Les constantes FEXE *% indiquent la façon dont s'effectue l'ouverture d'une console ou le démarrage d'un programme.

Fichier NSFEXE.NCL

Les verbes de la librairie NSFexe, décrits ci après, sont déclarés dans le fichier texte écrit en NCL, de nom NSFEXE.NCL. Ce fichier peut aussi contenir des verbes complémentaires (API non publique). Vous pouvez donc désirer consulter ce fichier pour avoir la référence exhaustive de la librairie.

Pour consulter le fichier NSFEXE.NCL :

1. Placez vous dans le répertoire <NATSTAR>\NCL ou <NSDK>\NCL
<NATSTAR> représente le répertoire que vous avez choisi au moment de l'installation de NatStar et <NSDK> celui de NS-DK.
2. Editez le fichier NSFEXE.NCL avec n'importe quel éditeur de texte.

Catégories fonctionnelles de la librairie NSFexe

Ouvrir et fermer des consoles DOS ou OS/2

Fonction **OPENCONSOLE%** (Librairie NSFexe)

Ouvre une console et retourne son handle.

Syntaxe	OPENCONSOLE% (<i>titre, création, handle-fenêtre, num-event, mode, posx, posy, largeur, hauteur</i>)			
Paramètres	titre	CSTRING	I	titre de la console
	création	INT(1)	I	création ou non d'une nouvelle console lors de l'ouverture
	handle-fenêtre	INT(4)	I	handle de la fenêtre recevant les notifications
	num-event	INTEGER	I	numéro de l'événement utilisateur utilisé pour l'envoi des notifications
	mode	INTEGER	I	mode de lancement de la console
	posx	INTEGER	I	
	posy	INTEGER	I	position de la console
	largeur	INTEGER	I	
	hauteur	INTEGER	I	taille de la console

Valeur retournée	INT(4) 0 : erreur lors de l'ouverture de la console non nul : handle de la console
-------------------------	--

1. titre n'est affiché que lorsque la console apparaît sous la forme d'une fenêtre non maximisée. Il figure alors dans la barre de titre de cette dernière.
2. Si création vaut FALSE%, une console déjà existante est utilisée. Si aucune console n'existe, OPENCONSOLE% en crée une. Si création vaut TRUE%, une nouvelle console est créée.
3. Les notifications envoyées par la console (constantes CON NFY *%) sont redirigées vers la fenêtre identifiée par handle-fenêtre. Elles sont reçues dans l'événement utilisateur dont le numéro est indiqué dans num-event (0 pour l'événement USER0, 1 pour l'événement USER1, et ainsi de suite jusqu'à 15). Les notifications sont envoyées dans les cas suivants :
 - a) informations envoyées par le programme et affichées dans la console (celles-ci peuvent être reçues ligne par ligne ou en bloc selon le mode de réception indiqué dans STARTCONSOLEEXEC%),
 - b) terminaison du programme lancé dans la console,
 - c) terminaison de la console elle-même.
 - d) Aucune notification n'est envoyée si 0 est passé dans win-handle.
4. mode doit être égal à l'une des constantes FEXE *%. Ce paramètre sert à préciser la taille de la console lors de son ouverture : normale, maximisée, minimisée, etc.
5. posx, posy, largeur et hauteur précisent l'emplacement et la taille de la console lors de son ouverture. Ils ne sont significatifs que si mode vaut FEXE USEPOSITIONSIZE%.

Exemple :

```

LOCAL INT HD_Cons%(4)
LOCAL INTEGER Errcode%
LOCAL INTEGER Retcode%

; Crée une nouvelle console avec demande de notification dans l'événement USER 2 de la
; fenêtre courante
HD_Cons% = OPENCONSOLE%("Console title" ,TRUE%, SELF%, 2, FEXE_NORMAL%, 0, 0, 0, 0)
IF HD_Cons% <> 0
; Démarrage d'un programme en synchrone
IF STARTCONSOLEEXEC%(HD_Cons%, "APP.EXE", "", "." TRUE%, TRUE%, TRUE%,ErrCode%,
RetCode%)
; Le retour de STARTCONSOLEEXEC% signifie la
; terminaison du message lancé en synchrone
; Affichage de son code retour
MESSAGE "Programme terminé","Code retour =" && RetCode%
ELSE
MESSAGE "Lancement du programme impossible", "Erreur système =" && ErrCode%
ENDIF

```

```

IF NOT CLOSECONSOLE%(HD_Cons%)
MESSAGE "Attention", "Erreur lors de la fermeture de la console"
ENDIF
ELSE
MESSAGE "Attention", " Erreur lors de l'ouverture de la console "
ENDIF

```

Voir aussi [STARTCONSOLEEXEC%](#), [STOPCONSOLEEXEC%](#), [CLOSECONSOLE%](#)

Fonction CLOSECONSOLE% (Librairie NSFexe)

Ferme une console.

Syntaxe	CLOSECONSOLE% (handle-console)		
Paramètre	handle-console	INT(4)	handle de la console à fermer
Valeur retournée	INT(1) TRUE% : la console a été fermée FALSE% : erreur lors de la fermeture de la console		

La console doit avoir été ouverte par un appel à [OPENCONSOLE%](#) qui retourne le handle de la console.

Exemple :

```
; voir l'exemple illustrant OPENCONSOLE%
```

Voir aussi [OPENCONSOLE%](#), [constantes CON_NFY_*](#)

Constantes CON_NFY_*% (Librairie NSFexe)

Types de notifications envoyées par une console dans un événement utilisateur.

Syntaxe	Déclaration interne	Description
CON_NFY_CLOSED%	5	indique que la boîte a été fermée soit par un appel à CLOSECONSOLE% , soit par l'utilisateur (menu système de la boîte).

CON_NFY_PROGENDED%	3	indique que le programme qui était exécuté dans la boîte est arrêté. PARAM34% contient le code de sortie du programme.
CON_NFY_PROGOUTPUT%	4	indique un envoi d'informations du programme. PARAM2\$ est alors une chaîne CSTRING contenant le message.

1. L'événement utilisateur est l'événement spécifié lors de l'ouverture de la console par OPENCONSOLE%.
2. Le type de notification figure dans le paramètre PARAM12% de l'événement USERx, éventuellement accompagné d'information complémentaire passée dans PARAM34% ou PARAM2\$.
3. La seule notification possible pour un programme lancé avec STARTEXECUTEEX% est EXEC NFY TERMINATE%.

Voir aussi OPENCONSOLE%, CLOSECONSOLE%

Modes d'ouverture de console et de démarrage des programmes (constantes)

Constantes FEXE_ *% (Librairie NSFexe)

Façon dont s'effectue l'ouverture d'une console ou le démarrage d'un programme.

Syntaxe	Déclaration interne	Description
FEXE_HIDDEN%	\$0001	la console ou la fenêtre principale du programme est cachée lors de son ouverture.
FEXE_MAXIMIZED%	\$0002	la console ou la fenêtre principale du programme est maximisée lors de son ouverture.
FEXE_MINIMIZED%	\$0004	la console ou la fenêtre principale du programme est minimisée lors de son ouverture.

FEXE_NORMAL%	\$0000	la console ou la fenêtre principale du programme est ouverte avec une taille et une position déterminées par le système.
FEXE_USEPOSITIONSIZE%	\$8000	la console ou la fenêtre principale du programme est ouverte avec les dimensions précisées dans l'appel à la fonction <u>OPENCONSOLE%</u> .

Un programme peut ne pas se conformer à ces indications si son concepteur n'a pas prévu de tenir compte de ces données lors de son démarrage.

Voir aussi OPENCONSOLE%, STARTEXCUTEEX%

Démarrer / arrêter un programme sur une console

Fonction **STARTCONSOLEEXECEX%** (Librairie NSFexe)

Lance l'exécution d'un programme (exécutable ou traitement batch) dans une console.

Syntaxe	STARTCONSOLEEXECEX% (<i>handle-console, nom-programme, paramètres, répertoire, mode, écho, synchro, code-err, code-retour</i>)			
Paramètres	handle-console	INT(4)	I	handle de la console dans laquelle le programme va s'exécuter
	nom-programme	CSTRING	I	nom du programme
	paramètres	CSTRING	I	paramètres de lancement du programme
	répertoire	CSTRING	I	répertoire de travail du programme
	mode	INT(1)	I	mode de réception des informations envoyées par le programme
	écho	INT(1)	I	affichage ou non des informations dans la console

	synchro	INT(1)	I	lancement synchrone ou asynchrone
	code-err	INTEGER	I/O	code d'erreur système
	code-retour	INTEGER	I/O	code retour du programme
Valeur retournée	INT(1) TRUE% : le programme a été lancé FALSE% : erreur lors du lancement du programme. code-err contient le numéro d'erreur système qui identifie le problème.			

1. Le programme va s'exécuter dans la console ouverte par OPENCONSOLE% et identifiée par handle-console.
2. répertoire est le répertoire de travail du programme nom-prog. C'est l'équivalent de l'information portée dans le champ Répertoire de la boîte Propriétés du gestionnaire de programme de Windows.
3. mode est le mode de réception de la sortie du programme : soit global (mode vaut alors FALSE%), soit ligne à ligne (mode vaut TRUE%).
4. Selon la valeur de écho (TRUE% ou FALSE%), la sortie du programme va apparaître ou non dans la console.
5. Le programme peut être lancé en synchrone (synchro = TRUE%) ou en asynchrone (synchro = FALSE%) :
 - a) Lorsqu'un programme est lancé en synchrone:
 - i. l'application ayant effectué le lancement est bloquée jusqu'à la terminaison du programme. Ce blocage s'applique aux entrées clavier et souris mais pas aux messages qui sont toujours pris en compte.
 - ii. le code retour renvoyé par le message lors de sa terminaison est reçu dans code-retour
 - b) Lorsqu'un programme est lancé en asynchrone :
 - i. le programme s'exécute en parallèle avec l'application ayant effectué le lancement
 - ii. seule une notification permet de savoir que le programme se termine, code-retour n'est donc pas significatif dans ce cas-là.
6. La variable code-err ne doit être testée que lorsque STARTCONSOLEEXECEX% retourne FALSE% sinon sa valeur n'est pas significative. La variable code-retour n'est significative que pour un programme lancé en synchrone et retournant une valeur lors de sa terminaison. Par convention, le programme doit prévoir de renvoyer la valeur 0 (zéro) pour indiquer sa bonne terminaison, toute autre valeur est spécifique au programme.

7. Notification et code retour sont synonymes pour un programme lancé en synchrone, le message de notification étant reçu par l'application avant de pouvoir lire le code retour. La lecture de ce code par l'une ou l'autre façon ne dépend que du développeur de l'application.

Exemple :

```
LOCAL INTEGER Errcode%
LOCAL INTEGER Retcode%

; Crée une nouvelle console avec demande de notification dans l'événement USER 2 de la
fenêtre courante
HD_Cons% = OPENCONSOLE("Console title" ,TRUE%, SELF%, 2, FEXE_NORMAL%, 0, 0, 0, 0)
IF HD_Cons% <> 0
; Démarrage d'un programme en synchrone
IF STARTCONSOLEEXEC%(HD_Cons%, "APP.EXE", "", "." TRUE%, TRUE%, TRUE%,ErrCode%,
RetCode%)
; Le retour de STARTCONSOLEEXEC% signifie la terminaison du message lancé en synchrone
; Affichage de son code retour
MESSAGE "Programme terminé","Code retour =" && RetCode%
ELSE
MESSAGE "Lancement du programme impossible", "Erreur système =" && ErrCode%
ENDIF
IF NOT CLOSECONSOLE%(HD_Cons%)
MESSAGE "Attention", "Erreur lors de la fermeture de la console"
ENDIF
ELSE
MESSAGE "Attention", " Erreur lors de l'ouverture de la console "
ENDIF
```

Voir aussi STOPCONSOLEEXEC%

Fonction **STOPCONSOLEEXEC%** (Librairie NSFexe)

Arrête le programme lancé dans une console.

Syntaxe	STOPCONSOLEEXEC% (<i>handle-console</i>)		
Paramètre	handle-console	INT(4)	handle de la console où le programme s'exécute
Valeur retournée	INT(1) TRUE% : le programme a été arrêté FALSE% : erreur lors de l'arrêt du programme		

Le programme doit avoir été démarré par un appel à STARTCONSOLEEXEC% .

Exemple :

```
;Événement EXECUTED du bouton 'Start'
LOCAL INTEGER Errcode%
LOCAL INTEGER Retcode%
```

```

; Crée une nouvelle console avec demande de notification dans l'événement USER 2 de la
fenêtre courante
HD_Cons% = OPENCONSOLE("Titre Console" ,TRUE%, SELF%, 2, FEXE_NORMAL%, 0, 0, 0, 0)
IF HD_Cons% <> 0
; Démarrage d'un programme en asynchrone
IF STARTCONSOLEEXECEX%(HD_Cons%, "APP.EXE", "", "" TRUE%, TRUE%, FALSE%,ErrCode%,
RetCode%)
...
ELSE
MESSAGE "Lancement du programme impossible", "Erreur système =" && ErrCode%
ENDIF
ELSE
MESSAGE "Attention", " Erreur lors de l'ouverture de la console "
ENDIF
Evénement EXECUTED du bouton 'Stop'
IF STOPCONSOLEEXEC% (HD_Cons%)
MESSAGE "Console","Le programme s'est terminé normalement"
ELSE
MESSAGE "Attention","Erreur lors de la terminaison du programme"
ENDIF

```

Voir aussi [STARTCONSOLEEXECEX%](#)

Démarrer / arrêter un programme

Fonction **STARTEXCUTEEX%** ([Librairie NSFexe](#))

Démarre un programme et retourne son identificateur.

Syntaxe	STARTEXCUTEEX% (<i>nom-programme, paramètre, répertoire, handle-fenêtre, num-event, synchro, code-err, code-retour, mode, posx, posy, largeur, hauteur</i>)			
Paramètres	nom-programme	CSTRING	I	nom du programme
	paramètre	CSTRING	I	paramètres de lancement du programme
	répertoire	CSTRING	I	répertoire de travail du programme
	handle-fenêtre	INT(4)	I	handle de la fenêtre recevant les notifications
	num-event	INTEGER	I	numéro de l'événement utilisateur utilisé pour l'envoi des notifications.
	synchro	INT(1)	I	lancement synchrone ou asynchrone

	code-err	INTEGER	I/O	code erreur système
	code-retour	INTEGER	I/O	code retour du programme
	mode	INTEGER	I	mode d'affichage du programme
	posx	INTEGER	I	
	posy	INTEGER	I	position de la fenêtre principale du programme
	largeur	INTEGER	I	
	hauteur	INTEGER	I	taille de la fenêtre principale du programme
Valeur retournée	INTEGER 0 : erreur lors du lancement du programme. code-err contient le numéro d'erreur système qui identifie le problème. non nul : identificateur du programme qui a été lancé			

1. répertoire est le répertoire de travail du programme nom-programme. C'est l'équivalent de l'information portée dans le champ Répertoire de la boîte Propriétés du gestionnaire de programme de Windows.

2. Les informations du programme sont redirigées vers la fenêtre identifiée par handle-fenêtre. Elles sont reçues dans l'événement utilisateur dont le numéro est indiqué dans num-event (0 pour l'événement USER0, 1 pour l'événement USER1, et ainsi de suite jusqu'à 15).

La notification n'est envoyée que lorsque le programme se termine, qu'il soit lancé en synchrone ou en asynchrone. Dans ce cas, PARAM12% du message utilisateur vaut EXEC NFY TERMINATE% et PARAM34% contient le code retour du programme. Notification et code-retour sont synonymes pour un programme lancé en synchrone, le message de notification étant reçu par l'application avant de pouvoir lire le code retour. La lecture de ce code par l'une ou l'autre façon ne dépend que du développeur de l'application.

Aucune notification n'est envoyée si 0 est passé dans win-handle. Cela peut-être le cas, par exemple, si vous désirez de lire le code retour du programme dans la variable return-code.

3. Le programme peut être lancé en synchrone (synchro = TRUE%) ou en asynchrone (synchro = FALSE%) :

a) Lorsqu'un programme est lancé en synchrone

- i. l'application ayant effectué le lancement est bloquée jusqu'à la terminaison du programme. Ce blocage s'applique aux entrées clavier et souris mais pas aux messages qui sont toujours pris en compte.
- ii. le code retour éventuellement renvoyé par le message lors de sa terminaison est reçu dans code-retour,
- b) Lorsqu'un programme est lancé en asynchrone:
 - i. le programme s'exécute en parallèle avec l'application ayant effectué le lancement,
 - ii. seule une notification permet de savoir que le programme se termine, code-retour n'est donc pas significatif dans ce cas-là.
- 4. La variable code-err ne doit être testée que lorsque STARTCONSOLEEXECEX% retourne FALSE% sinon sa valeur n'est pas significative. La variable code-retour n'est significative que pour un programme lancé en synchrone et retournant une valeur lors de sa terminaison. Par convention, le programme doit prévoir de renvoyer la valeur 0 (zéro) pour indiquer sa bonne terminaison, toute autre valeur est spécifique au programme.
- 5. mode doit être égal à l'une des constantes FEXE *%. Elle indique la taille de la fenêtre principale du programme lors de son affichage : normale, maximisée, minimisée, etc.

Un programme peut ne pas se conformer à ces indications si son concepteur n'a pas prévu de tenir compte de ces données lors de son démarrage. Pour un programme réalisé avec NatStar (ou NS-DK), la fenêtre principale doit être paramétrée en "shell position" pour prendre en compte ce mode. Une position et une taille précises peuvent être indiquées à l'aide de poss, posy, largeur et hauteur lorsque mode vaut FEXE USEPOSITIONSIZE%. Ces valeurs doivent être exprimées en pixels.

Exemple :

```

Événement EXECUTED du bouton 'Start'
ProgID% = STARTEXCUTEEX("APP.EXE" , "", "",SELF%, 2, FALSE%,ErrCode%,
RetCode%,FEXE_NORMAL%,0,0,0,0)
Événement USER2 de la fenêtre
IF PARAM12% = EXEC_NFY_TERMINATE%
MESSAGE "Le programme s'est terminé ","Code retour =" && PARAM34%
ENDIF
  
```

Voir aussi STOPEXCUTEEX, STARTEXCUTE2%

Fonction STARTEXCUTE2% (Librairie NSFexe)

Lance l'exécution du programme nom prog, avec les paramètres params, dont les messages seront envoyés dans l'événement utilisateur de numéro num msg (0

pour l'événement USER0, 1 pour l'événement USER1, etc.) de la fenêtre de handle fen.

répertoire-travail est le répertoire de travail du programme nom-prog. C'est l'équivalent de l'information portée dans le champ Répertoire de la boîte Propriétés du gestionnaire de programme de Windows.

Selon la valeur de synch, le programme sera exécuté en asynchrone (synch vaut alors FALSE%) ou en synchrone (synch vaut TRUE%) ; dans ce deuxième cas, le code qui suit l'appel à STARTEXCUTE2% ne sera effectué qu'une fois le programme terminé.

Si le lancement du programme s'est correctement déroulé, STARTEXCUTE2% retourne un entier, représentant l'identification de la session démarrée. Dans le cas contraire, STARTEXCUTE2% retourne 0. En cas de problème lors du lancement du programme, erreur retour est le numéro d'erreur explicitant ce problème.

Syntaxe	STARTEXCUTE2% (nom prog, params, répertoire-travail, handle fen, num msg, synch, erreur retour)			
Paramètres	nom prog	CSTRING		nom du programme
	params	CSTRING		paramètres
	répertoire-travail	CSTRING		répertoire de travail du programme
	handle fen	INT(4)		handle de la fenêtre recevant les notifications
	num msg	INTEGER		numéro de l'événement utilisateur
	synch	INT(1)		synchrone/asynchrone
	erreur retour	INTEGER		!V! erreur système
Valeur retournée	INTEGER			

Exemple :

```
;Événement EXECUTED du bouton 'Start'
Local INTEGER MyErrCode%
Local INT IDSession%

IDSession% = STARTEXCUTE2%("APP.EXE", "", "c:\", self%, 2, False%, MyErrCode%)
Événement USER2 de la fenêtre
IF PARAM1% = CON_NFY_PROGOUTPUT%
...
```

```
ELSE
STOPEXECUTE2 IDSession%
ENDIF
```

Voir aussi [STARTEXCUTEEX%](#), [STOPEXCUTE2](#)

Instruction STOPEXCUTEEX ([Librairie NSFexe](#))

Demande l'arrêt d'un programme.

Syntaxe	STOPEXCUTEEX <i>ID-programme</i>		
Paramètre	ID-programme	INTEGER	Identificateur du programme à arrêter

1. Le programme doit avoir été démarré par un appel à [STARTEXCUTEEX%](#).
2. Envoie le message système de fermeture ('close') au programme identifié par ID-programme. Le programme peut alors gérer sa propre interruption, en proposant, par exemple, une sauvegarde des modifications avant fermeture.

La demande d'arrêt du programme n'est pas envoyée si la fenêtre principale de ce dernier n'est pas active, c'est le cas notamment lorsque la fenêtre active du programme est une boîte de dialogue modale (comportement identique au Task Manager).

3. Un programme conçu avec NatStar (ou NS-DK) reçoit alors un événement CHECK qui permet d'indiquer (RETURN 0 ou 1) si le programme se termine effectivement ou non.
4. Identique à [KILLEXCUTE](#)

Exemple :

```
;Événement EXECUTED du bouton 'Start'
ProgID% = STARTEXCUTEEX("APP.EXE" , "", "",SELF%, 2, FALSE%,ErrCode%,
RetCode%,FEXE_NORMAL%,0,0,0,0)
Événement EXECUTED du bouton 'Stop'
STOPEXCUTEEX ProgID%
;Événement CHECK de la fenêtre principale du programme
; Vérifie si l'utilisateur veut réellement arrêter le programme
IF ASK2("Warning","Do you really want to quit APP ?")= YES%
; Acceptation: fermeture de l'application
RETURN 0
ELSE
; Refus: pas de fermeture de l'application
RETURN 1
ENDIF
```

Voir aussi [STARTEXCUTEEX%](#)

Instruction **STOPEXECUTE2** (Librairie NSFexe)

Envoie le message système de fermeture ('close') au programme lancé dans la session *Id session*. Le programme peut alors gérer sa propre interruption, en proposant, par exemple, une sauvegarde des modifications avant fermeture. Cette session a été préalablement démarrée par un appel à STARTEXECUTE2%, qui avait retourné ce numéro de session.

Syntaxe	STOPEXECUTE2 <i>Id session</i>		
Paramètre	Id session	INTEGER	Id de la session à arrêter

Exemple :

```
;Événement EXECUTED du bouton 'Start'
IDSession% = STARTEXECUTE2%("APP.EXE","", SELF%, 2, FALSE%)
Événement USER2 de la fenêtre
IF PARAM1% = CON_NFY_PROGOUTPUT
...
ELSE
STOPEXECUTE2 IDSession%
ENDIF
```

Voir aussi STARTEXECUTE2%, STOPEXECUTEEX

Instruction **KILLEXECUTE** (Librairie NSFexe)

Arrête un programme.

Syntaxe	KILLEXECUTE <i>ID-program</i>		
Paramètre	ID-program	INTEGER	Identificateur du programme à arrêter

1. Le programme doit avoir été démarré par un appel à STARTEXECUTEEX% qui retourne l'identificateur du programme.
2. Un programme arrêté avec KILLEXECUTE n'a aucun moyen d'empêcher sa terminaison à la différence de STOPEXECUTEEX qui effectue une demande préalable au programme.

Exemple :

```
;Événement EXECUTED du bouton 'Start'
ProgID% = STARTEXECUTEEX%("APP.EXE" , "", "",SELF%, 2, FALSE%,ErrCode%,
```

```
RetCode%,FEXE_NORMAL%,0,0,0,0)
Événement USER2 de la fenêtre
IF PARAM12% = EXEC_NFY_TERMINATE%
MESSAGE "Le programme s'est terminé ","Code retour =" && PARAM34%
ENDIF
Événement EXECUTED du bouton 'Stop'
; Arrête immédiatement le programme
KILLEXECUTE ProgID%
```

Voir aussi [STOPEXECUTEEX](#), [STARTEXECUTEEX%](#)

Constante EXEC_NFY_TERMINATE% (Librairie NSFexe)

Syntaxe	Déclaration interne	Description
EXEC_NFY_TERMINATE%	0	Indication de terminaison de programme.

1. Cette constante est envoyée dans le paramètre PARAM12% de l'événement utilisateur spécifié lors du lancement du programme par [STARTEXECUTEEX%](#). Elle notifie à la fenêtre NatStar (ou NS-DK) la terminaison du programme. Cette constante est la seule notification qui peut être envoyée par un programme démarré par [STARTEXECUTEEX%](#).
2. Le paramètre PARAM34% de l'événement USERx contient le code de sortie du programme.

Voir aussi [STARTEXECUTEEX%](#)

LIBRAIRIE DE FONCTIONS DIVERSES NSMISC

Cette librairie (Misc = Miscellaneous) contient de nombreuses fonctions et instructions variées :

- gestion de fichiers,
- gestion de disques (répertoires, place disponible, recherche de fichiers),
- génération de nombres aléatoires,
- conversions ASCII/EBCDIC,
- etc.

Installation

Déclarez NSMISC.NCL dans les librairies nécessaires au développement de votre application.

Vérifiez que le fichier NSxxMISC.DLL est bien dans un des répertoires PATH sous Windows.

Fichier NSMISC.NCL

Les verbes de la librairie NSMisc, décrits ci-après, sont déclarés dans le fichier texte écrit en NCL, de nom NSMISC.NCL. Ce fichier peut aussi contenir des verbes complémentaires (API non publique). Vous pouvez donc désirer consulter ce fichier pour avoir la référence exhaustive de la librairie.

Pour consulter le fichier NSMISC.NCL :

1. Placez-vous dans le répertoire <NATSTAR>\NCL ou <NSDK>\NCL
<NATSTAR> représente le répertoire que vous avez choisi au moment de l'installation de NatStar et <NSDK> celui de NS-DK.
2. Editez le fichier NSMISC.NCL avec n'importe quel éditeur de texte.

Catégories fonctionnelles de la librairie NSMisc

Démarrage d'une autre application

Instruction EXECPROG (Librairie NSMisc)

Démarre une application.

Syntaxe	EXECPROG <i>nom-fichier-exe, chaîne-param, mode, posx, posy, largeur, hauteur</i>
----------------	--

Paramètres	nom-fichier-exe	CSTRING	I	nom du fichier exécutable à démarrer
	chaîne-param	CSTRING	I	paramètres à rajouter à la ligne de commande
	mode	INTEGER	I	mode de démarrage de l'application
	posx	INTEGER	I	position x de la fenêtre principale de l'application
	posy	INTEGER	I	position y de la fenêtre principale de l'application
	largeur	INTEGER	I	largeur de la fenêtre principale de l'application
	hauteur	INTEGER	I	hauteur de la fenêtre principale de l'application

1. Le paramètre mode doit être une combinaison des constantes MODE *%, et sert à préciser la façon de démarrer l'application : normale, maximisée, iconisée, etc....
2. EXECPROG permet de démarrer un EXE généré avec NatStar (ou NS-DK) en MODE_MINIMIZE% ou en MODE_MAXIMIZE%. Pour cela, il faut que la fenêtre principale (Main Window) soit en "Shell Position & Size" et avec une bordure retaillable (Size Border). Il ne permet pas de démarrer un EXE généré en MODE_INVISIBLE%.
3. Testez MISCERROR% pour savoir si l'application a pu être démarrée avec succès. Les codes d'erreur possibles sont donnés dans l'annexe B de ce manuel.
4. Posx, posy, largeur, hauteur ne sont pas significatifs sous Windows.
5. L'application (mère) qui fait l'EXECPROG et l'application (fille) démarrée par EXECPROG fonctionnent simultanément, grâce aux multi-tâches de Windows. Les exécutables sont indépendants : l'application fille n'est pas arrêtée si l'application mère se termine.

6. Les programmes démarrés par EXECPROG sont des exécutables Windows ou DOS, des fichiers .COM, .BAT ou .PIF. Le passage de paramètres n'est accepté que par les fichiers exécutables .EXE. Tout programme autre qu'un exécutable Windows est lancé en fonction des indications fournies dans le fichier .PIF associé ou du fichier .PIF par défaut. De même, la terminaison du programme (fermeture de la fenêtre DOS dans lequel il a été lancé, par exemple) doit être spécifiée dans le fichier .PIF associé.

Exemple 1 :

```
; APP.EXE est une application mode graphique conçue avec NatStar
EXECPROG "C:\APP\APP.EXE", "", MODE_NORMAL%, 0, 0, 0, 0
IF MISCERROR% = 0
MESSAGE "OK", "Application APP.EXE démarrée !"
ELSE
MESSAGE "Erreur" && MISCERROR%, "Application APP.EXE non démarrée"
ENDIF
```

Exemple 2 :

```
; Démarrage du NOTEPAD en minimisé
EXECPROG "C:\WINDOWS\notepad.exe", "", MODE_MINIMIZE%, 0, 0, 0, 0
```

Exemple 3 :

```
; Démarrage d'un fichier de commandes .CMD (une application caractère comme CMD.EXE lancée
avec MODE_VIO_SCREEN% est exécutée de façon visible)
; Attention. L'emploi de /K au lieu de /C laisse la session CMD.EXE ouverte après
l'exécution de BATCH.CMD.
EXECPROG "C:\OS2\CMD.EXE", "/K BATCH.CMD", MODE_VIO_SCREEN%, 0, 0, 0, 0
```

Exemple 4 :

```
; Démarrage d'un éditeur fonctionnant en mode caractère
EXECPROG "C:\EDIT.EXE", "C:\FIC\INFO.TXT", MODE_NORMAL%, 0, 0, 0, 0
```

Voir aussi MODE *%, PARAMCOUNT%, PARAMSTR\$, STARTEXECUTE2%, STARTCONSOLEEXEC% (*librairie NSFEXE*)

Constantes MODE_* (Librairie NSMisc)

Valeurs définissant la façon de démarrer une application.

Syntaxe	Déclaration interne	Description
MODE_FULL_SCREEN%		précise que l'application est en plein écran.
MODE_INVISIBLE%	1	démarre l'application cachée (elle devra elle-même faire un SHOW).

MODE_MAXIMIZE%	2	démarre l'application maximisée.
MODE_MINIMIZE%	4	démarre l'application minimisée (i.e. icônisée).
MODE_NOAUTOCLOSE%		
MODE_NORMAL%	0	démarre l'application de façon "normale", sans forcer un état, une position ou une taille. L'application démarrée peut être un exécutable Windows ou DOS, un fichier .COM, .BAT ou .PIF.
MODE_PM_SCREEN%	0	précise que l'application est en mode graphique, de type Presentation Manager.
MODE_USE_POSITION%	32768	indique d'utiliser les positions et tailles passées avec EXECPROG.
MODE_VIO_SCREEN%		

1. Ces constantes permettent de définir comment démarre l'application lancée avec EXECPROG. Ne pas confondre avec les constantes COMMAND_*% de la librairie NSWin.
2. MODE_INVISIBLE% n'est pas implémenté pour les applications générées avec NatStar (ou NS-DK).
3. Le comportement lié aux autres modes dépendent du système sous lequel s'exécute l'application. MODE_NOAUTOCLOSE%, MODE_USE_POSITION%, MODE_PM_SCREEN%, MODE_FULL_SCREEN% et MODE_VIO_SCREEN% ne sont pas supportés sous Windows.
4. La façon de démarrer un programme autre qu'un exécutable Windows doit être préalablement spécifié dans le fichier .PIF associé. Si un tel programme a été démarré dans une fenêtre, le comportement de la fenêtre suite à son arrêt doit également être indiqué dans le fichier .PIF.
5. Certaines constantes peuvent se combiner : par exemple MODE_PM_SCREEN% + MODE_USE_POSITION% fonctionne, mais d'autres n'ont aucun intérêt : par exemple, MODE_FULL_SCREEN% + MODE_USE_POSITION%.

Voir aussi [EXECPROG](#)

Fonction **PARAMCOUNT%** (Librairie NSMisc)

Retourne le nombre de paramètres passés sur la ligne de commande lors du démarrage de l'application.

Syntaxe	PARAMCOUNT%
Valeur retournée	INTEGER

1. Si PARAMCOUNT% retourne 0, l'application a été démarrée sans aucun paramètre.
2. Supposons une application APP.EXE. Si cette application est démarrée depuis la ligne de commande du menu File/Run (Fichier/Exécuter) de Windows de la façon suivante :

```
APP NAT SYSTEM
```

alors PARAMCOUNT% retourne 2.

3. Le nom et les paramètres de lancement de l'application sont obtenus avec [PARAMSTR\\$](#).

Exemple :

```
; Affichage du nom de l'application et des éventuels paramètres
MESSAGE "Nom de l'application", PARAMSTR$(0)
MOVE 1 TO I%
WHILE I% <= PARAMCOUNT%
MESSAGE "Paramètre" && I%, PARAMSTR$(I%)
MOVE I%+1 TO I%
ENDWHILE
```

Voir aussi [PARAMSTR\\$](#), [EXECPROG](#), [RemoveCmdLineParam](#), [HideRemovedCmdLineParams%](#)

Fonction **PARAMSTR\$(index)** (Librairie NSMisc)

Retourne un paramètre de lancement de l'application.

Syntaxe	PARAMSTR\$(index)		
Paramètre	index	INTEGER	index du paramètre
Valeur retournée	CSTRING		

1. PARAMSTR\$(0) retourne le nom de l'application. Le premier paramètre de lancement a pour index 1. Le nombre total de paramètres de lancement est égal à [PARAMCOUNT%](#).

2. Supposons une application APP.EXE. Si cette application est démarrée depuis la ligne de commande du menu File/Run (Fichier/Exécuter) de Windows de la façon suivante :

APP NAT SYSTEM

Alors :
PARAMSTR\$(0) retourne "APP",
PARAMSTR\$(1) retourne "NAT",
PARAMSTR\$(2) retourne "SYSTEM".
Exemple :

```
; Affichage du nom de l'application et des éventuels paramètres
MESSAGE "Nom de l'application", PARAMSTR$(0)
MOVE 1 TO I%
WHILE I% <= PARAMCOUNT%
MESSAGE "Paramètre" && I%, PARAMSTR$(I%)
MOVE I%+1 TO I%
ENDWHILE
```

Voir aussi [PARAMCOUNT%](#), [EXECPROG](#) , [RemoveCmdLineParam](#), [HideRemovedCmdLineParams%](#)

Instruction RemoveCmdLineParam (Librairie NSMisc)

Retire (cache) le paramètre de ligne de commande Parm%. Le nombre total de paramètres de lancement (correspondant au résultat de la fonction PARAMCOUNT%) diminue de 1 et le paramètre suivant prend sa place dans la liste de la fonction PARAMSTR\$() et ainsi de suite.

Syntaxe	RemoveCmdLineParam Parm%		
Paramètre	Parm%	INTEGER	Paramètre de lancement (de 1 à PARAMCOUNT%)

- 1. On ne peut pas retirer la ligne PARAMSTR\$(0), car PARAMSTR\$(0) retourne le nom de l'application. Le premier paramètre de lancement a pour index 1
- 2. Le nombre total de paramètres de lancement est égal à PARAMCOUNT%.

Voir aussi [PARAMCOUNT%](#), [PARAMSTR%](#), [HideRemovedCmdLineParams%](#)

Fonction HideRemovedCmdLineParams%(Librairie NSMisc)

Permet de révéler tous les paramètres ayant été cachés.

Syntaxe	HideRemovedCmdLineParams% Hidden%
---------	-----------------------------------

Paramètre	Hidden%	INTEGER	I	Booléen (FALSE% ou TRUE%)
Valeur retournée	INT(1)			

1. Pour dévoiler les paramètres cachés par l'instruction RemoveCmdLineParam, positionner le paramètre Hidden% à FALSE%. Pour les cacher de nouveau, positionner le paramètre Hidden% à TRUE%.
2. Cette instruction permet de récupérer l'ancien état des paramètres de lancement pour pouvoir sauvegarder l'état actif lors d'un premier appel pour le changer.

Exemple :

```
bak% = HideRemovedCmdLineParams%(FALSE%)
;puis restaurer l'état actif quand on a terminé d'utiliser PARAMCOUNT% et PARAMSTR$ dans
l'état modifié
HideRemovedCmdLineParams%(bak%)
```

Voir aussi [PARAMCOUNT%](#), [PARAMSTR%](#), [RemoveCmdLineParam](#)

Fonction MakeCmdLineParam\$ (Librairie NSMisc)

Permet de récupérer une chaîne de caractères dans son état originel après avoir été passé en paramètre de ligne de commande dans un autre programme.

Syntaxe	MakeCmdLineParam\$ (Param\$)			
Paramètre	Param\$	CSTRING	I	chaîne de caractères
Valeur retournée	CSTRING			

1. La fonction MakeCmdLineParam\$ prend une chaîne de caractère et la reconditionne si besoin est pour que si elle est passée à un autre programme comme paramètre de ligne de commande, un appel à PARAMSTR\$ dans cet autre programme retournera la chaîne originale (avant reconditionnement).
2. Si la chaîne d'origine contient un ou des espaces, celle retournée par MakeCmdLineParam\$ sera entre doubles quotes (pour ne pas être considérée comme plusieurs paramètres séparés sur la ligne de commande). Si elle contient des doubles quotes celles-ci seront précédées d'un anti-slash (\), et des doubles quotes précédées de n anti-slashes seront alors précédées de 2xn+1 anti-slashes. Enfin, si la chaîne originale se termine par n anti-slash et qu'elle doit être mise entre doubles quotes, il y aura alors 2xn anti-slashes avant la quote de fin. Les anti-slashes ne précédant pas une double quote ne sont pas modifiés. Par exemple, un programme A qui veut lancer un programme B (via EXECPROG, STARTEXECUTEEX%, STARTEXECUTE2%, STARTCONSOLEEXEC%,

STARTCONSOLEEXECEX%...) en lui repassant son PARAMSTR\$(2) sans se préoccuper de ce qu'il contient doit en fait passer MakeCmdLineParam\$(PARAMSTR\$(2)). Exemples de transformations :

Chaîne originale	Résultat de MakeCmdLineParam\$
abc	abc
abc def	"abc def"
\abc\def\	\abc\def\
\abc\def \"ghi\	"\abc\def \\"ghi\""

Voir aussi PARAMSTR\$, HideRemovedCmdLineParams%, RemoveCmdLineParam

Environnement DOS

Fonction FGETSIZEEX% (Librairie NSMisc)

Retourne la taille d'un fichier.

Syntaxe	FGETSIZEEX% (nom-fichier)		
Paramètre	nom-fichier	CSTRING	I nom du fichier
Valeur retournée	INT(8) 0 : le fichier n'existe pas ou n'a pas été trouvé non nulle : taille du fichier en octets		

La valeur retournée étant un int(8) elle n'est plus limité à la capacité maximale d'un INT(4) environ 2 Giga octets.

Le fichier est recherché dans le répertoire courant si nom-fichier ne contient pas de chemin d'accès.

Exemple :

```
MESSAGE "Taille de CONFIG.SYS", FGETSIZEEX%("C:\CONFIG.SYS")
```

Voir aussi DISKSIZE%, F_SIZE%, FGETSIZE%

Fonction ENVCOUNT% (Librairie NSMisc)

Retourne le nombre de variables d'environnement déclarées dans la session DOS.

Syntaxe	ENVCOUNT%
Valeur retournée	INTEGER

Une variable d'environnement se déclare grâce à la commande SET de DOS.

Exemple :

```
; Affichage de toutes les variables d'environnement
MOVE 1 TO I%
WHILE I% <= ENVCOUNT%
MESSAGE "Variable" && I%, ENVSTR$(I%)
MOVE I%+1 TO I%
ENDWHILE
```

Voir aussi [GETLONGENV\\$](#), [ENVSTR\\$](#), [GETENV\\$](#), [PUTENV](#)

Fonction ENVSTR\$ (Librairie NSMisc)

Retourne la variable d'environnement dont l'index est passé en paramètre.

Syntaxe	ENVSTR\$ (index)		
Paramètre	index	INTEGER	index de la variable d'environnement
Valeur retournée	CSTRING		

1. La première variable d'environnement a pour index 1.
2. Le nombre total de variables d'environnement est donné par la fonction [ENVCOUNT%](#).

Exemple :

```
; Affichage de toutes les variables d'environnement
MOVE 1 TO I%
WHILE I% <= ENVCOUNT%
MESSAGE "Variable" && I%, ENVSTR$(I%)
MOVE I%+1 TO I%
ENDWHILE
```

Voir aussi [GETLONGENV\\$](#), [ENVCOUNT%](#), [GETENV\\$](#), [PUTENV](#)

Fonction GETENV\$ (Librairie NSMisc)

Retourne la valeur d'une variable d'environnement.

Syntaxe	GETENV\$ (env-string)
----------------	------------------------------

Paramètre	env-string	CSTRING		variable d'environnement
Valeur retournée	CSTRING			

Exemple :

```
MESSAGE "Variable PATH", GETENV$("PATH")
MESSAGE "Variable NS-INI", GETENV$("NS-INI")
```

Voir aussi [GETLONGENV\\$](#), [ENVCOUNT%](#), [ENVSTR\\$](#), [PUTENV](#)

Fonction GETLONGENV\$ (Librairie NSMisc)

Retourne la valeur d'une variable d'environnement.

Syntaxe	GETLONGENV\$ (env-string)			
Paramètre	env-string	CSTRING		variable d'environnement
Valeur retournée	DYNSTR			

Cette fonction retourne des chaînes de caractères de taille non déterminée. Il est donc souvent plus pratique d'utiliser cette fonction au lieu de GETENV\$.

Exemple :

```
MESSAGE "Variable PATH", GETLONGENV$("PATH")
MESSAGE "Variable NS-INI", GETLONGENV$("NS-INI")
```

Voir aussi [GETENV\\$](#), [ENVCOUNT%](#), [ENVSTR\\$](#), [PUTENV](#)

Fonction LGETENV (Librairie NSMisc)

Retourne un pointeur sur la chaîne C contenant la variable d'environnement spécifiée.

Syntaxe	LGETENV (name\$)			
Paramètre	name\$	CSTRING		variable d'environnement
Valeur retournée	POINTER			

1. Contrairement à la fonction [GETENV\\$](#), la fonction LGETENV n'est pas limitée à 255 caractères.
2. En NCL, on doit utiliser un segment pour lire la chaîne.

Exemple :

```
; Exemple pour lire jusqu'à 1023 caractères du PATH
SEGMENT SEG_STR1023
CSTRING S(1023)
END SEGMENT ; SEG_STR1023
LOCAL PATH$(1023)
PATH$=SEG_STR1023(LGETENV("PATH")).S
```

Voir aussi GETENV\$, ENVCOUNT%, ENVSTR\$, PUTENV

Instruction PUTENV (Librairie NSMisc)

Permet d'affecter une valeur à une variable d'environnement.

Syntaxe	PUTENV <i>name\$, value\$</i>		
Paramètres	name\$	CSTRING	variable d'environnement
	value\$	CSTRING	valeur de la variable d'environnement

Voir aussi GETLONGENV\$, GETENV\$

Gestion simple de fichiers

Instruction FERASE (Librairie NSMisc)

Détruit un fichier.

Syntaxe	FERASE <i>nom-fichier</i>		
Paramètre	nom-fichier	CSTRING	nom du fichier à détruire

Cette instruction est équivalente à la commande DEL ou ERASE de DOS.

Exemple 1 :

```
FERASE "TEST.TXT" ; détruit le fichier TEST.TXT situé dans le répertoire courant
```

Exemple 2 :

```
MOVE "\\NATSTAR\SAMPLES\TEST.SCR" TO S$
FERASE S$
```

Voir aussi RMDIR

Instruction **FRENAME** (Librairie NSMisc)

Renomme un fichier.

Syntaxe	FRENAME <i>ancien-nom-fichier, nouveau-nom-fichier</i>			
Paramètres	ancien-nom-fichier	CSTRING		ancien nom du fichier
	nouveau-nom-fichier	CSTRING		nouveau nom du fichier

Cette instruction est équivalente à la commande REN ou RENAME de DOS.

Exemple :

```
; Dans le répertoire courant, renomme HELLO.TXT en BONJOUR.TXT
FRENAME "HELLO.TXT", "BONJOUR.TXT"
```

Voir aussi **FCOPYFILE**

Instruction **FCOPYFILE** (Librairie NSMisc)

Copie un fichier sous un autre nom ou dans un autre fichier.

Syntaxe	FCOPYFILE <i>nom-fichier-source, nom-fichier-destination</i>			
Paramètres	nom-fichier-source	CSTRING		nom du fichier à copier
	nom-fichier-destination	CSTRING		nom du fichier destinataire

1. Cette instruction ne prend en compte qu'un fichier source. Elle n'est donc que partiellement équivalente à la commande COPY de DOS : les caractères '*' ou '?' ne sont pas reconnus.
2. Le fichier destinataire est créé s'il n'existe pas, sinon son contenu est entièrement remplacé par celui du fichier source.
3. Cette copie s'applique à tout type de fichier, aussi le résultat peut être lu indifféremment dans T_ERROR%, F_ERROR%.
4. Quel que soit le type d'erreur (le répertoire du fichier destination n'existe pas, pas assez de place sur le disque, etc), une seule valeur est retournée par les fonctions indiquées en 3 : ERROR_FAILED_TO_COPY_FILE% (voir annexe B de ce manuel).

5. Utilisez FRENAME si vous ne voulez que renommer un fichier.

Exemple :

```
; dans le répertoire courant, copie le contenu du fichier HELLO.TXT dans le fichier
GOODBYE.TXT situé dans le même répertoire
FCOPYFILE "HELLO.TXT","GOODBYE.TXT" ;
IF F_ERROR% <> 0
    BEEP
    MESSAGE "Erreur","Copie impossible : code d'erreur : " && F_ERROR%
ENDIF
```

Voir aussi FRENAME

Fonction FGETATTR% (Librairie NSMisc)

Retourne les attributs d'un fichier.

Syntaxe	FGETATTR% (nom-fichier)		
Paramètre	nom-fichier	CSTRING	I nom du fichier
Valeur retournée	INTEGER		

Les attributs sont spécifiés par une combinaison de constantes FIND *%.

Exemple :

```
MOVE "C:\WINDOWS" TO F$
IF FGETATTR%(F$) BAND FIND_DIRECTORY% <> 0
MESSAGE F$, "est un répertoire"
ENDIF
```

Voir aussi FSETATTR, FFINDFIRST\$, FFINDNEXT\$, FIND *%

Instruction FSETATTR (Librairie NSMisc)

Modifie les attributs d'un fichier.

Syntaxe	FSETATTR nom-fichier, attributs			
Paramètres	nom-fichier	CSTRING	I	nom du fichier à modifier
	attributs	INTEGER	I	nouveaux attributs de fichier

Les attributs de fichier sont précisés par une combinaison de constantes FIND *%.

Exemple :

```
; Cache et protège contre l'écriture le fichier TEST.TXT  
FSETATTR "TEST.TXT", FIND_READONLY% + FIND_HIDDEN%
```

Voir aussi FGETATTR%, FFINDFIRST\$, FFINDNEXT\$, FIND *%

Fonction FGETSIZE% (Librairie NSMisc)

Retourne la taille d'un fichier.

Syntaxe	FGETSIZE% (nom-fichier)		
Paramètre	nom-fichier	CSTRING	I nom du fichier
Valeur retournée	INT(4) 0 : le fichier n'existe pas ou n'a pas été trouvé non nulle : taille du fichier en octets		

La valeur retournée étant un int(4) elle est limitée à sa capacité maximale, soit environ 2 Giga octets.

Pour connaître la taille des fichiers de taille supérieure, utiliser FGETSIZEEX%

Le fichier est recherché dans le répertoire courant si nom-fichier ne contient pas de chemin d'accès.

Exemple :

```
MESSAGE "Taille de CONFIG.SYS", FGETSIZE%("C:\CONFIG.SYS")
```

Voir aussi DISKSIZE%, F SIZE%, FGETSIZEEX%

Fonction FGETSIZEEX% (Librairie NSMisc)

Retourne la taille d'un fichier.

Syntaxe	FGETSIZEEX% (nom-fichier)		
Paramètre	nom-fichier	CSTRING	I nom du fichier
Valeur retournée	INT(8) 0 : le fichier n'existe pas ou n'a pas été trouvé		

	non nulle : taille du fichier en octets
--	---

La valeur retournée étant un int(8) elle n'est plus limitée à la capacité maximale d'un INT(4) environ 2 Giga octets.

Le fichier est recherché dans le répertoire courant si nom-fichier ne contient pas de chemin d'accès.

Exemple :

```
MESSAGE "Taille de CONFIG.SYS", FGETSIZEEX%("C:\CONFIG.SYS")
```

Voir aussi DISKSIZE%, F SIZE%, FGETSIZE%

Fonction FEXPAND\$ (Librairie NSMisc)

Retourne une chaîne représentant le chemin d'accès complet d'un fichier.

Syntaxe	FEXPAND\$ (filename)		
Paramètre	filename	CSTRING	nom du fichier
Valeur retournée	CSTRING		

1. FEXPAND\$ ne vérifie pas l'existence du fichier, elle concatène le nom du répertoire courant (nom du volume disque inclus) au nom de fichier passé en paramètre.
2. Si nom-fichier est une chaîne vide (""), FEXPAND\$ retourne une chaîne contenant le volume disque et le répertoire courant suivi de \.
3. Si nom-fichier est une chaîne ne contenant que le nom du fichier sans répertoire, FEXPAND\$ retourne une chaîne contenant le volume disque et le répertoire courant suivi du nom du fichier.

Exemple 1 :

```
MESSAGE "Le répertoire courant est :", FEXPAND$("")
```

Exemple 2 :

```
; Si le répertoire courant est D:\NATSTAR\SAMPLES
MOVE FEXPAND$("../TEST.TXT") TO S$
; Alors, S$ vaut "D:\NATSTAR\TEST.TXT"
```

Voir aussi FGETDIR\$, FGETTEXT\$, FGETNAME\$, GETDIR\$, GETPATHNAME\$ (librairie NSPATH)

Fonction FGETDIR\$ (Librairie NSMisc)

Retourne le répertoire d'un fichier à partir de son nom complet.

Syntaxe	FGETDIR\$ (nom-fichier)		
Paramètre	nom-fichier	CSTRING	I nom du fichier
Valeur retournée	CSTRING		

1. nom-fichier doit être le nom complet du fichier (volume, répertoire, nom fichier)
2. Cette fonction n'accède jamais au disque. Elle fonctionne uniquement à partir de la chaîne de caractères passée en paramètre dont elle extrait la partie correspondant au répertoire.
3. Si le répertoire de fichier n'est pas indiqué dans nom-fichier, FGETDIR\$ retourne une chaîne vide quel que soit l'emplacement réel du fichier sur le disque.
4. De la même manière, le nom du fichier et son extension sont obtenus avec FGETNAME\$ et FGETTEXT\$.
5. Ainsi FGETDIR\$(F\$) & FGETNAME\$(F\$) & FGETTEXT\$(F\$) est toujours strictement égal à F\$, quel que soit F\$.

Exemple :

```
MOVE "C:\NATSTAR\SCREENS\SAMPLE.SCR" TO F$  
MESSAGE "Répertoire", FGETDIR$(F$) ; affiche "C:\NATSTAR\SCREENS\"
```

Voir aussi FGETTEXT\$, FGETNAME\$, GETDIR\$, FEXPAND\$, GETPATHNAME\$ (librairie NSPATH)

Fonction FGETTEXT\$ (Librairie NSMisc)

Retourne l'extension d'un fichier à partir de son nom complet.

Syntaxe	FGETTEXT\$ (nom-fichier)		
Paramètre	nom-fichier	CSTRING	I nom du fichier
Valeur retournée	CSTRING		

1. nom-fichier doit être le nom complet du fichier (volume, répertoire, nom fichier)

2. Cette fonction n'accède jamais au disque. Elle fonctionne uniquement à partir de la chaîne de caractères passée en paramètre dont elle extrait la partie correspondant à l'extension du fichier.
3. De la même manière, le répertoire et le nom du fichier sont obtenus avec FGETDIR\$ et FGETNAME\$.
4. Ainsi FGETDIR\$(F\$) & FGETNAME\$(F\$) & FGETEXT\$(F\$) est toujours strictement égal à F\$, quel que soit F\$.

Exemple :

```
MOVE " C:\NSNATSTAR\SCREENS\SAMPLE.SCR " TO F$
MESSAGE " Extension ", FGETEXT$(F$) ; affiche " .SCR "
```

Voir aussi FGETDIR\$, FGETNAME\$, GETDIR\$, FEXPAND\$, GETPATHNAME\$ (librairie NSPATH)

Fonction FGETNAME\$ (Librairie NSMisc)

Retourne le nom d'un fichier à partir de son nom complet.

Syntaxe	FGETNAME\$ (nom-fichier)		
Paramètre	nom-fichier	CSTRING	I nom du fichier
Valeur retournée	CSTRING		

1. nom-fichier doit être le nom complet du fichier (volume, répertoire, nom fichier)
2. Cette fonction n'accède jamais au disque. Elle fonctionne uniquement à partir de la chaîne de caractères passée en paramètre dont elle extrait la partie correspondant au nom du fichier.
3. De la même manière, le répertoire et l'extension du nom du fichier sont obtenus avec FGETDIR\$ et FGETEXT\$.
4. Ainsi FGETDIR\$(F\$) & FGETNAME\$(F\$) & FGETEXT\$(F\$) est toujours strictement égal à F\$, quel que soit F\$.

Exemple :

```
MOVE "C:\NATSTAR\SCREENS\SAMPLE.SCR" TO F$
MESSAGE "Nom", FGETNAME$(F$) ; affiche "SAMPLE"
```

Voir aussi FGETDIR\$, FGETEXT\$, GETDIR\$, FEXPAND\$, GETPATHNAME\$ (librairie NSPATH)

Fonction FFINDFIRST\$ (Librairie NSMisc)

Retourne le nom du premier fichier ayant des attributs donnés.

Syntaxe	FFINDFIRST\$ (<i>nom-fichier, attributs</i>)			
Paramètres	nom-fichier	CSTRING		nom du fichier
	attributs	INTEGER		attributs du fichier
Valeur retournée	CSTRING			

1. Les caractères * et ? sont acceptés dans la chaîne nom-fichier.
2. Les attributs sont spécifiés par une combinaison de constantes FIND *%.
3. FFINDFIRST\$ retourne une chaîne vide lorsqu'aucun fichier n'a été trouvé.
4. Reportez vous aux commentaires sur les constantes FIND *% pour connaître les différents attributs pouvant être spécifiés.
5. La recherche des fichiers suivants ayant les mêmes caractéristiques s'effectue avec FFINDNEXT\$.

Exemple 1 :

```
; Recherche d'un fichier ayant un nom précis
IF FFINDFIRST$("C:\TEST.TXT", FIND_ANYFILE%) = ""
MESSAGE "Attention", "Le fichier C:\TEST.TXT n'existe pas"
ENDIF
```

Exemple 2 :

```
; Recherche de la taille et de la date d'écriture d'un fichier
MOVE FFINDFIRST$("C:\CONFIG.SYS", FIND_ANYFILE% + FIND_RETURN_SIZE% + FIND_RETURN_DATE%) TO F$
; F$ contient, par exemple, la chaîne "CONFIG.SYS,2296,33259"
```

Voir aussi FFINDNEXT\$, FGETATTR%, FIND *%, FFINDFIRSTEX\$, GETPATHNAME\$ (librairie **NSPATH**)

Fonction **FFINDNEXT\$** (Librairie **NSMisc**)

Suite à une recherche fructueuse effectuée avec FFINDFIRST\$, retourne le nom du fichier suivant ayant les mêmes caractéristiques que le premier fichier trouvé.

Syntaxe	FFINDNEXT\$
Valeur retournée	CSTRING

1. FFINDNEXT\$ retourne une chaîne vide lorsque tous les fichiers ont été trouvés.

2. Reportez-vous aux commentaires sur les constantes FIND_* pour connaître les différents attributs pouvant être spécifiés.

3. Date et heure obtenues dans les informations supplémentaires (constantes FIND_RETURN_*) sont sous forme d'entier. Les fonctions DATE\$ et TIME\$ de la librairie NSDate permettent de les convertir en chaîne de caractères.

Exemple 1 :

```
; Recherche de tous les fichiers *.DLL dans le répertoire C:\WINDOWS
MOVE FFINDFIRST$("C:\WINDOWS\*.DLL",FIND_ANYFILE%) TO F$
WHILE F$ <> ""
MESSAGE "Fichier trouvé", F$
MOVE FFINDNEXT$ TO F$
ENDWHILE
```

Exemple 2 :

```
; Recherche des répertoires de la racine du volume C c'est à dire des fichiers ayant
uniquement l'attribut Directory positionné
MOVE FFINDFIRST$("C:\*.\"",FIND_DIRECTORY% + FIND_ORED_ONLY%) TO F$
WHILE F$ <> ""
MESSAGE "Répertoire trouvé", F$
MOVE FFINDNEXT$ TO F$
ENDWHILE
```

Voir aussi FFINDFIRST\$, FFINDFIRSTEX\$, FFINDNEXTEX\$, FFINDCLOSEEX, FGETATTR%, FIND_*, GETPATHNAME\$ (librairie NSPATH)

Constantes FIND_* (Librairie NSMisc)

Attributs et critères de recherche de fichier.

Syntaxe	Déclaration interne	Description
FIND_ANYFILE%	\$37	Tout type de fichier
FIND_ARCHIVE%	\$20	Attribut archive sous DOS (ne correspond pas à l'attribut Archive des répertoires sous l'explorateur Windows)
FIND_DIRECTORY%	\$10	Répertoire
FIND_HIDDEN%	\$02	Fichier caché (n'apparaît pas lors d'un DIR)
FIND_READONLY%	\$01	Fichier en lecture seule (écriture interdite)
FIND_SYSFILE%	\$04	Fichier système
FIND_ORED_ONLY%	\$0100	Recherche sur combinaison stricte

FIND_RETURN_ATTR%	\$0200	La recherche retourne l'attribut
FIND_RETURN_SIZE%	\$0400	La recherche retourne la taille
FIND_RETURN_DATE%	\$0800	La recherche retourne la date
FIND_RETURN_TIME%	\$1000	La recherche retourne l'heure

1. FIND_ANYFILE% à FIND_SYSFILE% sont les attributs de fichier utilisés par les fonctions FFINDFIRST\$, FGETATTR% et l'instruction FSETATTR.

2. FIND_ORED_ONLY% est utilisée par FFINDFIRST\$. Combinée avec les attributs de fichier, elle indique que la recherche ne porte que sur les fichiers dont les attributs sont strictement identiques à cette combinaison.

3. Les constantes FIND_RETURN_*% sont utilisées par FFINDFIRST\$. Combinées avec les attributs de fichier, elles indiquent quelles informations supplémentaires (attribut, taille, date écriture, heure écriture) doivent être retournées par FFINDFIRST\$ et FFINDNEXT\$ à la suite du nom de fichier.

4. Les constantes FIND_READONLY%, FIND_HIDDEN%, FIND_SYSFILE%, FIND_DIRECTORY% et FIND_ARCHIVE% peuvent se combiner (par exemple FIND_HIDDEN% + FIND_DIRECTORY% correspond à un répertoire caché). FIND_ANYFILE% est la somme des cinq premières constantes.

5. Le résultat d'une recherche inclut toujours les fichiers "normaux". Ainsi une recherche avec FIND_DIRECTORY% retourne les répertoires mais également les fichiers normaux. Une distinction stricte se fait en employant FIND_ORED_ONLY%.

6. Les constantes FIND_RETURN_*% permettent d'obtenir des informations supplémentaires sur le fichier recherché. Celles-ci sont données à la suite du nom de fichier retourné par FFINDFIRST\$ et FFINDNEXT\$. Les informations retournées figurent toujours dans l'ordre : attribut, taille, date écriture, heure écriture et sont séparées par une virgule. Elles ne sont fournies que lorsque la constante associée a été indiquée dans le paramètre attribut de FFINDFIRST\$.

Voir aussi FFINDFIRST\$, FGETATTR%, FSETATTR

Fonction FFINDFIRSTEX\$ (Librairie NSMisc)

Retourne le nom long du premier fichier ayant des attributs donnés.

Syntaxe	<u>FFINDFIRSTEX\$</u> (<i>path\$, attr\$, hdir%</i>)			
Paramètres	path\$	CSTRING		nom du fichier
	attr\$	INT		attributs du fichier

	hdir%	INT(4)	O	handle de recherche
Valeur retournée	CSTRING			

1. Les caractères * et ? sont acceptés dans la chaîne path\$.
2. Les attributs sont spécifiés par une combinaison de constantes FIND *%.
3. FFINDFIRSTEX\$ retourne une chaîne vide lorsqu'aucun fichier n'a été trouvé.
4. Le paramètre hdir% sert à conserver un handle de recherche. Il permet ainsi l'imbrication ou la récursion de plusieurs boucles FFINDFIRSTEX\$/FFINDNEXTEX\$. FFINDFIRSTEX\$ initialise cette variable qui doit être passée à FFINDNEXTEX\$ (dans une boucle) et à FFINDCLOSEEX (à la fin de la boucle).
5. Reportez-vous aux commentaires sur les constantes FIND *% pour connaître les différents attributs pouvant être spécifiés.
6. La recherche des fichiers suivants ayant les mêmes caractéristiques s'effectue avec FFINDNEXTEX\$.
7. L'instruction FFINDCLOSEEX doit impérativement être appelée à la fin de l'utilisation de FFINDFIRSTEX\$/FFINDNEXTEX\$.

Voir aussi FFINDNEXTEX\$ FFINDCLOSEEX FFINDFIRST\$ FFINDNEXT\$, FGETATTR%, FIND *%, GETPATHNAME\$ (librairie NSPATH)

Fonction FFINDNEXTEX\$ (Librairie NSMisc)

Suite à une recherche fructueuse effectuée avec FFINDFIRSTEX\$, retourne le nom long du fichier suivant ayant les mêmes caractéristiques que le premier fichier trouvé.

Syntaxe	FFINDNEXTEX\$ (hdir%)			
Paramètre	hdir%	INT	I	handle de recherche
Valeur retournée	CSTRING			

1. FFINDNEXTEX\$ retourne une chaîne vide lorsque tous les fichiers ont été trouvés.
2. Reportez-vous aux commentaires sur les constantes FIND *% pour connaître les différents attributs pouvant être spécifiés.
3. Date et heure obtenues dans les informations supplémentaires (constantes FIND RETURN *%) sont sous forme d'entier. Les fonctions DATE\$ et TIME\$ de la librairie NSDate permettent de les convertir en chaîne de caractères.

4. L'instruction FFINDCLOSEEX doit impérativement être appelée à la fin de l'utilisation de FFINDFIRSTEX\$/FFINDNEXTEX\$.

Voir aussi FFINDFIRSTEX\$ FFINDCLOSEEX FFINDFIRST\$ FFINDNEXT\$, FGETATTR%, FIND_*%, GETPATHNAME\$ (librairie NSPATH)

Instruction FFINDCLOSEEX (Librairie NSMisc)

Libère le handle d'énumération de nom de fichier.

Syntaxe	FFINDCLOSEEX <i>hdir%</i>		
Paramètre	<i>hdir%</i>	INT	I handle de recherche

L'instruction FFINDCLOSEEX doit impérativement être appelée à la fin de l'utilisation de FFINDFIRSTEX\$/FFINDNEXTEX\$.

Voir aussi FFINDFIRSTEX\$, FFINDNEXTEX\$

Instruction FGETTIME (Librairie NSMisc)

L'instruction FGETTIME retourne la date et l'heure d'un fichier dans les variables *year%*, *month%*, *day%*, *hour%*, *min%* et *sec%*.

Syntaxe	FGETTIME <i>file\$, year%, month%, day%, hour%, min%, sec%</i>			
Paramètres	<i>file\$</i>	CSTRING	I	Nom du fichier dont on veut connaître la date et l'heure.
	<i>year%</i>	INTEGER	O	Année
	<i>month%</i>	INTEGER	O	Mois
	<i>day%</i>	INTEGER	O	Jour
	<i>hour%</i>	INTEGER	O	Heure
	<i>min%</i>	INTEGER	O	Minutes
	<i>sec%</i>	INTEGER	O	Secondes

Si le fichier ne peut être lu, la fonction MISCERROR% renvoie la valeur ERROR_CANNOT_ACCESS_TO_FILE%.

Voir aussi FSETTIME

Instruction FSETTIME (Librairie NSMisc)

La fonction FSETTIME modifie la date et l'heure d'un fichier à partir des variables *year%*, *month%*, *day%*, *hour%*, *min%* et *sec%*.

Syntaxe	FSETIME <i>file\$, year%, month%, day%, hour%, min%, sec%</i>			
Paramètres	file\$	CSTRING	I	nom du fichier dont on veut modifier la date et l'heure.
	year%	INTEGER	I	Année
	month%	INTEGER	I	Mois
	day%	INTEGER	I	Jour
	hour%	INTEGER	I	Heure
	sec%	INTEGER	I	Secondes

Si le fichier ne peut être modifié, la fonction MISCERROR% renvoie la valeur ERROR_CANNOT_ACCESS_TO_FILE%

Voir aussi FGETTIME

Fonction NEXISTFILE% (Librairie NSMisc)

Cette fonction permet de tester l'existence d'un fichier sur un disque.

Syntaxe	NEXISTFILE% (<i>file</i>)			
Paramètre	file	CSTRING	I	nom d'un fichier
Valeur retournée	INT(2) TRUE% si le fichier existe et FALSE% s'il n'existe pas.			

Voir aussi NEXISTDIR%

Fonction NEXISTDIR% (Librairie NSMisc)

Cette fonction permet de tester l'existence d'un répertoire sur un disque.

Syntaxe	NEXISTDIR% (<i>file</i>)			
Paramètre	file	CSTRING	I	nom d'un répertoire
Valeur retournée	INT(2) TRUE% si le répertoire existe et FALSE% s'il n'existe pas.			

Voir aussi NEXISTFILE%

Gestion de répertoires et de disques

Instruction CHDIR (Librairie NSMisc)

Change de répertoire courant.

Syntaxe	CHDIR <i>nom-répertoire</i>		
Paramètre	nom-répertoire	CSTRING	I nom du nouveau répertoire courant

L'instruction CHDIR est équivalente à la commande CD ou CHDIR de DOS.

Exemple :

```
CHDIR ".." ; remonte au répertoire supérieur
CHDIR "TEST" ; descend dans le répertoire TEST situé sous le répertoire courant
```

Voir aussi CHDISK, MKDIR, RMDIR, GETDIR\$

Fonction GETDIR\$ (Librairie NSMisc)

Retourne le répertoire courant d'un disque donné.

Syntaxe	GETDIR\$ (<i>numéro-disque</i>)		
Paramètre	numéro-disque	INT(1)	I numéro de disque
Valeur retournée	CSTRING		

1. numéro-disque doit être compris entre 0 et 26 :

Valeur	Disque
0	courant
1	A:
2	B:
3	C:
4...25	D: ... Y:
26	Z:

2. GETDIR\$(0) et GETDIR\$(GETDISK%) sont équivalents, mais GETDISK% ne vaut jamais 0 !

Exemple :

```
MESSAGE "Le répertoire courant est", GETDIR$(0)
```

Voir aussi [CHDIR](#), [MKDIR](#), [RMDIR](#), [GETPATHNAME\\$](#) (librairie NSPATH)

Instruction MKDIR (Librairie NSMisc)

Crée un répertoire.

Syntaxe	MKDIR <i>nom-répertoire</i>		
Paramètre	nom-répertoire	CSTRING	nom du répertoire à créer

1. Cette instruction est équivalente à la commande MD ou MKDIR de DOS.
2. Si aucun chemin d'accès n'est précisé dans nom-répertoire, le répertoire créé est un sous-répertoire du répertoire courant.

Exemple 1 :

```
MKDIR "TEST" ; Crée un répertoire TEST situé sous le répertoire courant
```

Exemple 2 :

```
MOVE " C:\NATSTAR\SAMPLES " TO S$
MKDIR S$
```

Voir aussi [CHDIR](#), [RMDIR](#), [GETDIR\\$](#), [GETDISK%](#)

Instruction RMDIR (Librairie NSMisc)

Détruit un répertoire.

Syntaxe	RMDIR <i>directory-name</i>		
Paramètre	directory-name	CSTRING	nom du répertoire à détruire

1. Cette instruction est équivalente à la commande RD ou RMDIR de DOS.
2. Si le chemin d'accès au répertoire n'est pas précisé dans nom-répertoire, la destruction s'applique au sous-répertoire nom-répertoire du répertoire courant.

Exemple 1 :

```
RMDIR "TEST" ; Détruit le répertoire TEST situé sous le répertoire courant
```

Exemple 2 :

```
MOVE " \NATSTAR\SAMPLES " TO S$
RMDIR S$
```

Voir aussi CHDIR, MKDIR, GETDIR\$, GETDISK%, GETDISKS%

Instruction CHDISK (Librairie NSMisc)

Change de disque logique courant.

Syntaxe	CHDISK <i>numéro-disque</i>			
Paramètre	numéro-disque	INT(1)	I	numéro du nouveau disque courant

L'entier numéro-disque passé à CHDISK doit être compris entre 0 et 26 :

Valeur	Disque:
0	courant
1	A:
2	B:
3	C:
4 ... 25	D: ... Y:
26	Z:

Exemple :

```
CHDISK 5
; E devient le disque courant
IF MISCERROR% <> 0
MESSAGE "ERREUR","le disque logique demandé n'existe pas !"
ENDIF
```

Voir aussi GETDISK%, GETDISKS%

Fonction DISKFREE% (Librairie NSMisc)

Retourne la taille, en octets, disponible sur un disque.

Syntaxe	DISKFREE% (<i>numéro-disque</i>)			
Paramètre	numéro-disque	INT(1)	I	numéro du disque
Valeur retournée	INT(4)			

1. La fonction retourne -1 si le disque n'existe pas ou n'est pas accessible.
2. numéro-disque doit être un entier compris entre 0 et 26 :

Valeur	Disque
--------	--------

0	courant
1	A:
2	B:
3	C:
4 ... 25	D: ... Y:
26	Z:

3. La taille du disque est limitée à 2 GB.

Exemple :

```
IF DISKFREE%(0) > TAILLEMALISTE%
SAVE MALISTE TO "MALISTE.TXT"
ELSE
MESSAGE "Attention !", "Place insuffisante sur disque"
ENDIF
```

Voir aussi [DISKFREE#](#), [DISKSIZE%](#), [DISKSIZE#](#), [FGETSIZE%](#), [GETDIR\\$](#), [GETDISK%](#) , [DISKFREE#](#)

Fonction DISKFREE# (Librairie NSMisc)

Retourne la taille disponible, au format réel, sur le disque spécifié en paramètre.

Syntaxe	DISKFREE# (numéro-disque)		
Paramètre	numéro-disque	INT(1)	numéro du disque
Valeur retournée	REAL		

1. La fonction retourne -1 si le disque n'existe pas ou n'est pas accessible.
2. numéro-disque doit être un entier compris entre 0 et 26 :

Valeur	Disque
0	courant
1	A:
2	B:
3	C:
4 ... 25	D: ... Y:
26	Z:

3. La taille du disque n'est pas limitée à 2 GB contrairement à la fonction DISKFREE%.

Voir aussi [DISKFREE%](#), [DISKSIZE%](#), [DISKSIZE#](#), [FGETSIZE%](#), [GETDIR\\$](#), [GETDISK%](#)

Fonction **DISKSIZE%** ([Librairie NSMisc](#))

Retourne la taille totale, en octets, d'un disque.

Syntaxe	DISKSIZE% (<i>numéro-disque</i>)		
Paramètre	numéro-disque	INT(1)	numéro du disque
Valeur retournée	INT(4)		

1. La fonction retourne -1 si le disque n'existe pas ou n'est pas accessible.
2. numéro-disque doit être un entier compris entre 0 et 26 :

Valeur	Disque:
0	courant
1	A:
2	B:
3	C:
4 ... 25	D: ... Y:
26	Z:

3. La taille du disque est limitée à 2 GB.

Exemple :

```
MESSAGE "Capacité du disque C:", DISKSIZE%(3)
```

Voir aussi [DISKFREE%](#), [DISKFREE#](#), [DISKSIZE#](#), [FGETSIZE%](#), [GETDIR\\$](#), [GETDISK%](#), [DISKSIZE#](#)

Fonction **DISKSIZE#** ([Librairie NSMisc](#))

Retourne la taille totale, au format réel, d'un disque.

Syntaxe	DISKSIZE# (<i>numéro-disque</i>)		
Paramètre	numéro-disque	INT(1)	numéro du disque
Valeur retournée	REAL		

1. La fonction retourne -1 si le disque n'existe pas ou n'est pas accessible.
2. numéro-disque doit être un entier compris entre 0 et 26 :

Valeur	Disque:
0	courant
1	A:
2	B:
3	C:
4 ... 25	D: ... Y:
26	Z:

3. La taille du disque n'est pas limitée à 2 GB contrairement à la fonction DISKSIZE%.

Voir aussi DISKSIZE%, DISKFREE%, DISKFREE#, FGETSIZE%, GETDIR\$, GETDISK%, DISKSIZE%

Constantes DT_*% (Librairie NSMisc)

Les constantes DT_ % sont utilisées conjointement avec la fonction GETDISKTYPE pour connaître le type d'un disque.

Syntaxe	Déclaration interne	Description
DT_REMOVABLE%		Disque amovible ou disquette
DT_FIXED%		Disque fixe
DT_REMOTE%		Disque distant ou réseau
DT_CDROM%		Disque CD-ROM
DT_RAMDISK%		Disque RAM ou mémoire

Voir aussi GETDISKTYPE%, GETVOLUMENAMES\$

Fonction GETDISK% (Librairie NSMisc)

Retourne le numéro du disque courant.

Syntaxe	GETDISK%
Valeur retournée	INT(1)

Chaque numéro de disque est un entier compris entre 0 et 26 :

Valeur	Disque
0	courant
1	A:
2	B:

3	C:
4 ... 25	D: ... Y:
26	Z:

Exemple :

```
MESSAGE "Le disque courant est", CHR$(64 + GETDISK%)
```

Voir aussi GETDISKS%, DISKFREE%, DISKSIZE%

Fonction GETDISKS% (Librairie NSMisc)

Retourne un entier indiquant quels sont les disques logiques disponibles.

Syntaxe	GETDISKS%
Valeur retournée	INT(4)

Chaque bit de l'entier retourné par GETDISKS% représente un disque logique (bit de poids 1 pour A, bit de poids 2 pour B, et ainsi de suite). Un bit à 1 indique la présence du disque logique.

Exemple :

```
LOCAL I%,J%,LDISK%

; Affichage dans la List Box LB_DISK des disques logiques disponibles
MOVE GETDISKS% TO I%
MOVE 1 TO J%
MOVE ASC("A") TO LDISK%
REPEAT
IF (I% BAND J%) = J%
INSERT AT END CHR$(LDISK%) TO LB_DISK
ENDIF
MOVE J%*2 TO J%
MOVE LDISK%+1 TO LDISK%
UNTIL LDISK% = ASC("Z")
```

Voir aussi GETDISK%, DISKFREE%, DISKSIZE%

Fonction GETDISKTYPE% (Librairie NSMisc)

Cette fonction retourne le type d'un disque dont le numéro est passé en paramètre.

Syntaxe	GETDISKTYPE% (volume%)		
Paramètre	volume%	INT(1)	I numéro du disque

Valeur retournée	INTEGER Constante de type <u>DT</u> *% précisant le type de disque.
-------------------------	--

1. Chaque numéro de disque est un entier compris entre 0 et 26 :

Valeur	Disque
0	courant
1	A:
2	B:
3	C:
4 ... 25	D: ... Y:
26	Z:

2. Si le type de disque ne peut être trouvé, la fonction MISCERROR% renvoie la valeur ERROR_CANNOT_GET_DISK%.

Voir aussi Constantes DT *%, GETVOLUMENAME\$

Fonction GETVOLUMENAME\$ (Librairie NSMisc)

Cette fonction retourne le label d'un disque dont le numéro est passé en paramètre.

Syntaxe	GETVOLUMENAME\$ (<i>volume%</i>)		
Paramètre	<i>volume%</i>	INT(1)	I numéro du disque
Valeur retournée	INTEGER Constante de type <u>DT</u> *% précisant le type de disque		

Chaque numéro de disque est un entier compris entre 0 et 26 :

Valeur	Disque
0	courant
1	A:
2	B:
3	C:
4 ... 25	D: ... Y:
26	Z:

Si le type de disque ne peut être trouvé, la fonction MISCERROR% renvoie la valeur ERROR_CANNOT_GET_DISK%.

Voir aussi Constantes DT *%, GETDISKTYPE%

Fonctions concatPath1..4\$ (Librairie nsMisc)

Concatène automatiquement des noms de répertoires et de fichier

concatPath1\$

Syntaxe	Function concatPath1\$ (dir\$, name\$)			
Paramètres	cString	dir\$	I	premier répertoire
	cString	name\$	I	nom du fichier
Valeur retournée	cString	nom complet du répertoire avec le nom de fichier.		

concatPath2\$

Syntaxe	Function concatPath2\$ (dir1\$, dir2\$, name\$)			
Paramètres	cString	dir1	I	premier répertoire
	cString	dir2	I	deuxième répertoire
	cString	name\$	I	nom du fichier
Valeur retournée	cString	nom complet du répertoire avec le nom de fichier.		

concatPath3\$

Syntaxe	Function concatPath3\$ (dir1\$, dir2\$, dir3\$, name\$)			
Paramètres	cString	dir1\$	I	premier répertoire
	cString	dir2\$	I	deuxième répertoire
	cString	dir3\$	I	troisième répertoire
	cString	name\$	I	nom du fichier
Valeur retournée	cString	nom complet du répertoire avec le nom de fichier.		

concatPath4\$

Syntaxe	Function concatPath4\$ (dir1\$, dir2\$, dir3\$, dir4\$, name\$)			
Paramètres	cString	dir1\$		premier répertoire
	cString	dir2\$		deuxième répertoire
	cString	dir3\$		troisième répertoire
	cString	dir4\$		quatrième répertoire
	cString	name\$		nom du fichier
Valeur retournée	cString	nom complet du répertoire avec le nom de fichier.		

Ces fonctions effectuent plusieurs actions.

1. choix du séparateur actif selon le système d'exploitation. (antiSlash pour Windows et slash pour Unix)
2. vérifie pour chacun des termes si le séparateur est présent ou pas au début et à la fin.
3. rajoute les séparateur intermédiaires, si nécessaire.
4. Supprime les séparateur intermédiaires, si un doublon est constaté.

Les fonction sont multipliées afin de pouvoir concaténer 2, 3, 4 ou 5 termes.

Remarque

Si le premier terme contient un indicateur de disque dur ('C:\', 'D:\' ...), il doit se terminer par un \ (antiSlash), car il n'est pas rajouté automatiquement dans ce cas, afin de permettre les chemins relatif.

Cet exemple crée un nom complet de fichier avec 5 termes

```
local s$
s$ = concatPath4$ ('c:\', 'projects\','test','sale','file.txt')
; donne
'c:\projects\test\sale\file.txt'
```

Note

On peut constater que l'addition ou la suppression de séparateurs est automatique.

Gestion d'erreur

Fonction MISCERROR% (Librairie NSMisc)

Retourne le code d'erreur de la dernière fonction ou instruction de la librairie NSMisc qui a été employée.

Syntaxe	MISCERROR%
Valeur retournée	INT(4)

1. Lorsqu'il n'y a pas eu d'erreur, MISCERROR% retourne zéro.
2. Les fonctions et instructions de gestions de fichiers F_* et T_* (sauf F_ERROR%) modifient la valeur retournée par MISCERROR%.
3. Voir également les différents codes d'erreur retournés par MISCERROR%.

Gestion de fichiers texte

Fonction T_APPEND% (Librairie NSMisc)

Ouvre un fichier texte ASCII en mode écriture avec le pointeur en fin de fichier.

Syntaxe	T_APPEND% (<i>nom-fichier</i>)		
Paramètre	nom-fichier	CSTRING	I nom du fichier à ouvrir
Valeur retournée	INT(4)		

1. Cette fonction permet de directement rajouter du texte en fin de fichier dès le premier T_WRITE suivant.
2. Des parenthèses utilisées au sein de la chaîne nom-fichier permettent d'utiliser les variables d'environnement :

```
; Ouvre le fichier SAMPLE.TXT situé dans le répertoire spécifié par la variable  
d'environnement NS-LST  
MOVE T_APPEND%("(NS-LST)SAMPLE.TXT") TO H%
```


3. Il est conseillé d'appeler `T_ERROR%` après chaque appel à une fonction ou instruction `T_*` de façon à vous assurer que cette dernière s'est bien déroulée.
Voir aussi `T_OPEN%`, `T_CLOSE`, Fonctions et Instructions `F_*`, Langage NCL : instructions `LOAD`, `SAVE`

Instruction `T_CLOSE` (Librairie NSMisc)

Ferme un fichier texte ASCII.

Syntaxe	<code>T_CLOSE handle-fichier</code>		
Paramètre	handle-fichier	INT(4)	handle du fichier à fermer

Les différents codes d'erreur retournés par `T_ERROR%` sont donnés dans [l'annexe B de ce manuel](#).

Exemple :

```
; Ecriture du fichier texte "TEST.TXT"
MOVE T_CREATE("TEST.TXT") TO H%
IF T_ERROR%<>0
MESSAGE "Erreur T_CREATE", "Impossible d'ouvrir le fichier"
EXIT
ENDIF

T_WRITELN H%, "Bonjour"
T_WRITELN H%, "Au revoir"
T_CLOSE H%
```

Voir aussi `T_APPEND%`, `T_CREATE%`, `T_EOF%`, `T_EOL%`, `T_ERROR%`, `T_OPEN%`, `T_READ%`, `T_READ#`, `T_READ$`, `T_READLN%`, `T_READLN#`, `T_READLN$`, `T_WRITE`, `T_WRITELN`, Fonctions et Instructions `F_*`, Langage NCL : Instructions `LOAD`, `SAVE`

Fonction `T_CREATE%` (Librairie NSMisc)

Crée un fichier texte ASCII en mode écriture et retourne son handle.

Syntaxe	<code>T_CREATE% (nom-fichier)</code>		
Paramètre	nom-fichier	CSTRING	nom du fichier à créer
Valeur retournée	INT(4)		

1. T_CREATE% ne crée le fichier texte que s’il n’existe pas, sinon T_CREATE% ouvre le fichier en écrasant son contenu.
2. Le handle retourné permet ensuite d’accéder au fichier à l’aide des autres fonctions et instructions T_+% de lecture/écriture.
3. Il est conseillé d’appeler T_ERROR% après chaque appel à une fonction ou instruction T_+% de façon à vous assurer que cette dernière s’est bien déroulée.
4. Des parenthèses utilisées au sein de la chaîne nom-fichier permettent d'utiliser les variables d'environnement :

```

; Crée le fichier SAMPLE.TXT situé dans le répertoire spécifié par la variable
d'environnement NS-LST
MOVE T_CREATE%("(NS-LST)SAMPLE.TXT") TO H%

```

Exemple :

```

; Ecriture du fichier texte "TEST.TXT"
MOVE T_CREATE%("TEST.TXT") TO H%
IF T_ERROR%<>0
MESSAGE "Erreur T_CREATE%", "Impossible d'ouvrir le fichier"
EXIT
ENDIF
T_WRITELN H%, "Bonjour"
T_WRITELN H%, "Au revoir"
T_CLOSE H%

```

Voir aussi T_APPEND%, T_CLOSE, T_EOF%, T_EOL%, T_ERROR%, T_OPEN%,T_READ*, T_WRITE*, Fonctions et Instructions F_*, Langage NCL : Instructions LOAD, SAVE

Fonction T_OPEN% (Librairie NSMisc)

Ouvre un fichier texte ASCII en mode lecture et retourne son handle.

Syntaxe	T_OPEN% (nom-fichier)		
Paramètre	nom-fichier	CSTRING	I nom du fichier à ouvrir
Valeur retournée	INT(4)		

1. Un fichier texte est un fichier constitué d’une suite de caractères imprimables. L’insertion de caractères CR-LF (Carriage Return - Line Feed) permet de le diviser en lignes de caractères telles qu’elles peuvent apparaître lorsque le fichier est lu à l’aide d’un éditeur de texte.
2. Le handle retourné permet ensuite d’accéder au fichier à l’aide des autres fonctions et instructions T_+% de lecture.

3. Lorsque le fichier est inexistant, T_OPEN% provoque une erreur. Celle-ci doit être lue à l'aide de T_ERROR%.
4. Des parenthèses utilisées au sein de la chaîne nom-fichier permettent d'utiliser les variables d'environnement :

```
; Ouvre le fichier SAMPLE.TXT situé dans le répertoire spécifié par la variable
d'environnement NS-LST
MOVE T_OPEN%("(NS-LST)SAMPLE.TXT") TO H%
```

Exemple :

```
; Lecture du fichier texte "TEST.TXT"
MOVE T_OPEN%("TEST.TXT") TO H%
IF T_ERROR%<>0
MESSAGE "Erreur T_OPEN%", "Impossible d'ouvrir le fichier"
EXIT
ENDIF

MOVE 0 TO I%
WHILE NOT T_EOF%(H%)
MOVE I%+1 TO I%
MESSAGE "Ligne numéro" && I%, T_READLN$(H%)
ENDWHILE
T_CLOSE H%
```

Voir aussi T_APPEND%, T_CLOSE, T_CREATE%, T_EOF%, T_EOL%, T_ERROR%, T_READ*, T_WRITE*

Instruction T_WRITE (Librairie NSMisc)

Ecrit une expression dans un fichier texte ASCII.

Syntaxe	T_WRITE <i>handle-fichier, expression</i>		
Paramètres	handle-fichier	INT(4)	handle de fichier
	expression	CSTRING	expression à écrire

1. L'expression est automatiquement convertie en chaîne de caractères.
2. Une fois l'écriture effectuée, le pointeur de position courante est positionné après le dernier caractère.
3. Si l'expression à écrire contient plus de 255 caractères, nous vous conseillons d'utiliser l'instruction T_WRITEEX.
4. T_WRITE retourne une erreur si le fichier a été ouvert en lecture avec T_OPEN%.
5. Il est conseillé d'appeler T_ERROR% après chaque appel à une fonction ou instruction T_*% de façon à vous assurer que cette dernière s'est bien déroulée.

Exemple :

```
; Le fichier est supposé avoir été créé et son handle copié dans la variable H%
T_WRITE H%, "Bonjour"
T_WRITE H%, "Au revoir"
; La ligne courante du fichier contient la chaîne
; "BonjourAu revoir"
T_CLOSE H%
```

Voir aussi [T_WRITEEX](#), [T_WRITELN](#), [T_WRITELNEX](#)

Instruction **T_WRITELN** ([Librairie NSMisc](#))

Ecrit une expression dans un fichier texte ASCII et ajoute un saut à la ligne suivante.

Syntaxe	T_WRITELN <i>handle-fichier, expression</i>		
Paramètres	handle-fichier	INT(4)	handle de fichier
	expression	CSTRING	expression à écrire

1. L'expression est automatiquement convertie en chaîne de caractères
2. Une fois l'écriture effectuée, le pointeur de position courante est positionné en début de ligne suivante.
3. Si l'expression à écrire contient plus de 255 caractères, nous vous conseillons d'utiliser l'instruction [T_WRITELNEX](#).
4. **T_WRITELN** retourne une erreur si le fichier a été ouvert en lecture avec [T_OPEN%](#).
5. Il est conseillé d'appeler [T_ERROR%](#) après chaque appel à une fonction ou instruction **T_*** de façon à vous assurer que cette dernière s'est bien déroulée.

Exemple :

```
; Le fichier est supposé avoir été créé et son handle copié dans la variable H%
T_WRITELN H%, "Bonjour"
T_WRITELN H%, "Au revoir"
; La dernière ligne du fichier contient la chaîne "Au revoir"
; et la ligne précédente la chaîne "Bonjour"
T_CLOSE H%
```

Voir aussi [T_WRITELNEX](#), [T_WRITE](#), [T_WRITEEX](#)

Instruction **T_WRITEEX** ([Librairie NSMisc](#))

Ecrit une expression dans un fichier texte ASCII.

Syntaxe	T_WRITEEX <i>handle-fichier, expression</i>
---------	--

Paramètres

- Voir aussi** *T WRITE*, *T WRITELN*, *T WRITELNEX*

Instruction T WRITELNEX (Librairie NSMisc)

Ecrit une expression dans un fichier texte ASCII et ajoute un saut à la ligne suivante.

Syntaxe

T WRITELNEX *handle-fichier, expression*

Paramètres	handle-fichier	INT(4)		handle de fichier
	expression	DYNSTR		expression à écrire

- Exemple :

```
; Le fichier est supposé avoir été créé et son handle copié dans la variable H%  
T_WRITELNEX H%,  
"Blablablablablablablablablablablablablablablablablablablablablablabla  
blablablablablablablablablablablablablablablablabla"
```

```
T_WRITELNEX H%, "Au revoir"
; La dernière ligne du fichier contient la chaîne "Au revoir" et la ligne précédente la
chaîne "Blabla..."
T_CLOSE H%
```

Voir aussi T_WRITELN, T_WRITE, T_WRITEEX

Fonction T_READ% (Librairie NSMisc)

Retourne un mot de la ligne courante d'un fichier de texte ASCII sous la forme d'un entier.

Syntaxe	<u>T_READ%</u> (<i>handle-fichier</i>)		
Paramètre	handle-fichier	INT(4)	handle du fichier
Valeur retournée	INT(4)		

1. Un mot est constitué de caractères consécutifs précédés par un début de ligne ou un caractère espace et suivi d'un caractère espace ou d'une fin de ligne. Le mot est pris en compte à partir de la position courante du curseur.
2. La valeur 0 (zéro) est retournée si le mot n'est pas convertible en entier.
3. Après la lecture, le curseur est positionné juste après le mot. Un nouveau T_READ% lit le mot suivant de la ligne ou le premier mot de la ligne suivante si la fin de la ligne courante avait été atteinte.
4. La conversion en réel est obtenue avec T_READ#
5. Il est conseillé d'appeler T_ERROR% après chaque appel à une fonction ou instruction T_* de façon à vous assurer que cette dernière s'est bien déroulée.

Exemple :

```
; Le fichier est supposé avoir été ouvert et son handle copié dans la variable H% chaque
ligne du fichier est supposée contenir les informations suivantes : âge salaire nom prénom
; par exemple, 24 8500.00 Potin Nathalie

; lecture de tout le fichier et affichage des informations
WHILE NOT T_EOF%(H%)
MESSAGE "Informations", "Age :" && T_READ%(H%) && "Salaire :" && T_READ#(H%) && "Nom :" &&
T_READLN$(H%)
ENDWHILE

T_CLOSE H%
```

Voir aussi T_READ#, T_READ\$, T_READLN%, T_READLN#, T_READLN\$

Fonction T_READ# (Librairie NSMisc)

Retourne un mot de ligne courante d'un fichier de texte ASCII sous la forme d'un réel.

Syntaxe	T_READ# (handle-fichier)		
Paramètre	handle-fichier	INT(4)	handle du fichier
Valeur retournée	NUM(8)		

1. Un mot est constitué de caractères consécutifs précédés par un début de ligne ou un caractère espace et suivi d'un caractère espace ou d'une fin de ligne. Le mot est pris en compte à partir de la position courante du curseur.
2. La valeur 0 (zéro) est retournée si le premier mot n'est pas convertible en réel.
3. Après la lecture, le curseur est positionné juste après le mot. Un nouveau T_READ# lit le mot suivant de la ligne ou le premier mot de la ligne suivante si la fin de la ligne courante avait été atteinte.
4. La conversion en entier est obtenue avec T_READ%.
5. Il est conseillé d'appeler T_ERROR% après chaque appel à une fonction ou instruction T_*% de façon à vous assurer que cette dernière s'est bien déroulée.

Exemple :

```
; Le fichier est supposé avoir été ouvert et son handle copié dans la variable H% chaque
ligne du fichier est supposée contenir les informations suivantes : âge salaire nom prénom
; par exemple, 24 8500.00 Potin Nathalie

; lecture de tout le fichier et affichage des informations
WHILE NOT T_EOF%(H%)
MESSAGE "Informations", "Age :" && T_READ%(H%) && "Salaire :" && T_READ#(H%) && "Nom :" &&
T_READLN$(H%)
ENDWHILE

T_CLOSE H%
```

Voir aussi T_READ%, T_READ\$, T_READLN%, T_READLN#, T_READLN\$

Fonction T_READ\$ (Librairie NSMisc)

Retourne le contenu de la ligne courante d'un fichier texte ASCII.

Syntaxe	T_READ\$ (handle-fichier)		
Paramètre	handle-fichier	INT(4)	handle du fichier
Valeur retournée	CSTRING(255)		

1. La lecture est effectuée à partir de la position courante du pointeur qui peut ne pas être en début de ligne si des lectures précédentes de type `T_READ%` ou `T_READ#` ont été faites auparavant.
2. La chaîne retournée contient au maximum 255 caractères, caractères de fin de lignes exceptés. Plusieurs `T_READ$` consécutifs devaient être effectués pour lire une ligne composée de plus de 255 caractères. La nouvelle fonction `T_READLNEX$` permet de contourner ce mécanisme pour les fichiers de plus de 255 caractères.
3. Même si la ligne entière a été lue, le pointeur de lecture n'est pas automatiquement positionné en début de ligne suivante par `T_READ$`. Dans le cas de lignes composées de plus de 255 caractères, il est nécessaire de tester si la fin de ligne a été atteinte afin d'effectuer une lecture avec `T_READLN$` pour forcer le passage à la ligne suivante. La nouvelle fonction `T_READLNEX$` permet de contourner ce mécanisme pour les fichiers de plus de 255 caractères.
4. Il est conseillé d'appeler `T_ERROR%` après chaque appel à une fonction ou instruction `T_*` de façon à vous assurer que cette dernière s'est bien déroulée.

Exemple :

Voir l'exemple illustrant `T_READ%`

Voir aussi `T_READ%`, `T_READ#`, `T_READLN%`, `T_READLN#`, `T_READLN$`

Fonction `T_READLN%` (Librairie `NSMisc`)

Retourne un mot de la ligne courante d'un fichier texte ASCII sous la forme d'un entier et positionne le pointeur de lecture en début de ligne suivante.

Syntaxe	<code>T_READLN%</code> (<i>handle-fichier</i>)		
Paramètre	handle-fichier	INT(4)	I handle du fichier
Valeur retournée	INT(4)		

1. La seule différence entre `T_READ%` et `T_READLN%` est que cette dernière instruction positionne le pointeur de lecture en début de ligne suivante immédiatement après la lecture.
2. Un mot est constitué de caractères consécutifs précédés par un début de ligne ou un caractère espace et suivi d'un caractère espace ou d'une fin de ligne. Le mot est pris en compte à partir de la position courante du curseur.
3. La valeur 0 (zéro) est retournée si le mot n'est pas convertible en entier.
4. Il est conseillé d'appeler `T_ERROR%` après chaque appel à une fonction ou instruction `T_*` de façon à vous assurer que cette dernière s'est bien déroulée.

Voir aussi T_READ%, T_READ#, T_READ\$, T_READLN#, T_READLN\$

Fonction T_READLN# (Librairie NSMisc)

Retourne un mot de la ligne courante d'un fichier rtexte ASCII sous la forme d'un réel et positionne le pointeur de lecture en début de ligne suivante.

Syntaxe	<u>T_READLN#</u> (<i>handle-fichier</i>)		
Paramètre	handle-fichier	INT(4)	I handle du fichier
Valeur retournée	NUM(8)		

1. La seule différence entre T_READ# et T_READLN# est que cette dernière fonction positionne le pointeur de lecture en début de ligne suivante immédiatement après la lecture.
2. Un mot est constitué de caractères consécutifs précédés par un début de ligne ou un caractère espace et suivi d'un caractère espace ou d'une fin de ligne. Le mot est pris en compte à partir de la position courante du curseur.
3. La valeur 0 (zéro) est retournée si le premier mot n'est pas convertible en réel.
4. Il est conseillé d'appeler T_ERROR% après chaque appel à une fonction ou instruction T_*% de façon à vous assurer que cette dernière s'est bien déroulée.

Voir aussi T_READ%, T_READ\$, T_READ#, T_READLN%, T_READLN\$

Fonction T_READLN\$ (Librairie NSMisc)

Retourne le contenu de la ligne courante d'un fichier texte ASCII et positionne le pointeur de lecture en début de ligne suivante.

Syntaxe	<u>T_READLN\$</u> (<i>handle-fichier</i>)		
Paramètre	handle-fichier	INT(4)	I handle du fichier
Valeur retournée	CSTRING		

1. La seule différence entre T_READ\$ et T_READLN\$ est que cette dernière fonction positionne le pointeur de lecture en début de ligne suivante immédiatement après la lecture.

2. La lecture est effectuée à partir de la position courante du pointeur qui peut ne pas être en début de ligne si des lectures précédentes de type `T_READ%` ou `T_READ#` ont été faites auparavant.
3. La chaîne retournée contient au maximum 255 caractères, caractères de fin de lignes exceptés. Plusieurs `T_READ$` consécutifs doivent être effectués pour lire une ligne composée de plus de 255 caractères. Si la chaîne retournée contient plus de 255 caractères, utilisez plutôt la fonction `T_READLNEX$`.
4. Il est conseillé d'appeler `T_ERROR%` après chaque appel à une fonction ou instruction `T_*` de façon à vous assurer que cette dernière s'est bien déroulée.

Exemple :

```
; Le fichier est supposé avoir été ouvert et son handle copié dans la variable H% chaque
ligne du fichier est supposée contenir les informations suivantes : âge salaire nom prénom
; par exemple, 24 8500.00 Potin Nathalie

; lecture de tout le fichier et affichage des informations
WHILE NOT T_EOF%(H%)
MESSAGE "Informations", "Age :" && T_READ%(H%) && "Salaire :" && T_READ#(H%) && "Nom :" &&
T_READLN$(H%)
ENDWHILE

T_CLOSE H%
```

Voir aussi `T_READ%`, `T_READ$`, `T_READ#`, `T_READLN%`, `T_READLN#`, `T_READLNEX$`

Fonction `T_READLNEX$` (Librairie `NSMisc`)

Retourne le contenu de la ligne courante d'un fichier texte ASCII et positionne le pointeur de lecture en début de ligne suivante.

Syntaxe	<code>T_READLNEX\$</code> (<i>handle-fichier</i>)		
Paramètre	handle-fichier	INT(4)	I handle du fichier
Valeur retournée	DYNSTR		

1. La seule différence entre `T_READLN$` et `T_READLNEX$` est que cette dernière fonction permet de retourner une chaîne contenant plus de 255 caractères.
2. La lecture est effectuée à partir de la position courante du pointeur qui peut ne pas être en début de ligne si des lectures précédentes de type `T_READ%` ou `T_READ#` ont été faites auparavant.
3. Il est conseillé d'appeler `T_ERROR%` après chaque appel à une fonction ou instruction `T_*` de façon à vous assurer que cette dernière s'est bien déroulée.

Voir aussi `T_READLN$`, `T_READ%`, `T_READ$`, `T_READ#`, `T_READLN%`, `T_READLN#`

Fonction T_EOF% (Librairie NSMisc)

Indique si le pointeur de position courante est situé en fin de fichier texte ASCII.

Syntaxe	T_EOF% (handle-fichier)		
Paramètre	handle-fichier	INT(4)	I handle du fichier
Valeur retournée	INT(1) TRUE% : Le pointeur de position courante est situé en fin de fichier. FALSE% : Le pointeur n'est pas situé en fin de fichier.		

Exemple :

```

; Lecture du fichier texte "TEST.TXT"
MOVE T_OPEN%("TEST.TXT") TO H%
IF T_ERROR%<>0
MESSAGE "Erreur T_OPEN%", "Impossible d'ouvrir le fichier"
EXIT
ENDIF

MOVE 0 TO I%
WHILE NOT T_EOF%(H%)
MOVE I%+1 TO I%
MESSAGE "Ligne numéro" && I%, T_READLN$(H%)
ENDWHILE
T_CLOSE H%

```

Voir aussi T_EOL%

Fonction T_EOL% (Librairie NSMisc)

Indique si le pointeur de position courante est situé en fin de ligne dans un fichier texte.

Syntaxe	T_EOL% (handle-fichier)		
Paramètre	handle-fichier	INT(4)	I handle du fichier
Valeur retournée	INT(1) TRUE% : Le pointeur de position courante est situé en fin de fichier. FALSE% : Le pointeur n'est pas situé en fin de fichier.		

Voir aussi T_EOF%

Fonction T_ERROR% (Librairie NSMisc)

Retourne le dernier code d'erreur dû à la gestion de fichiers texte.

Syntaxe	T_ERROR%
Valeur retournée	INT(4) 0 : Aucune erreur non nulle : Erreur dans la dernière fonction ou instruction T_+% utilisée. Voir également les différents <u>codes d'erreur</u> .

Il est conseillé d'appeler T_ERROR% après chaque appel à une fonction ou instruction T_+% de façon à vous assurer que cette dernière s'est bien déroulée.

Exemple :

Voir exemple illustrant T_OPEN%

Voir aussi T_APPEND%, T_CLOSE%, T_CREATE%, T_EOF%, T_EOL%, T_OPEN%, T_READ*, T_WRITE*

Gestion de fichiers binaires

Instruction F_CLOSE (Librairie NSMisc)

Ferme un fichier binaire.

Syntaxe	F_CLOSE <i>handle-fichier</i>		
Paramètre	handle-fichier	INT(4)	handle du fichier à mettre à jour

Le fichier binaire doit avoir été ouvert par F_OPEN% ou F_CREATE%.

Exemple :

Voir exemples illustrant F_CREATE% et F_OPEN%

Voir aussi F_CREATE%, F_OPEN%

Fonction F_CREATE% (Librairie NSMisc)

Crée un fichier binaire en mode lecture/écriture et retourne son handle.

Syntaxe	F_CREATE% (<i>taille-segment, nom-fichier</i>)
----------------	---

Paramètres	taille-segment	INTEGER	I	taille du segment utilisé au sein du fichier
	nom-fichier	CSTRING	I	nom du fichier
Valeur retournée	INT(4)			

1. Un fichier binaire est un fichier constitué d'une suite de segments NCL ayant toujours la même structure.
2. F_CREATE% crée un nouveau fichier. Si le fichier existe déjà, elle le supprime et en crée un nouveau.
3. La taille du segment peut être spécifiée aisément à l'aide de l'opérateur SIZEOF.
4. Le handle retourné permet ensuite d'accéder au fichier à l'aide des autres fonctions et instructions F_*% de lecture/écriture.
5. Des parenthèses utilisées au sein de la chaîne nom-fichier permettent d'utiliser les variables d'environnement :

```
; Crée le fichier SAMPLE.TXT situé dans le répertoire spécifié par la variable
d'environnement NS-LST
MOVE F_CREATE%(..., "(NS-LST)SAMPLE.TXT") TO H%
```

Exemple ;

```
; Déclaration du segment utilisé
; Événement INIT de la fenêtre
SEGMENT INDIVIDU
STRING NOM(31)
STRING PRENOM(31)
ENDSEGMENT

; Événement EXECUTED du bouton d'écriture dans le fichier
; Allocation mémoire "mappée" sur le segment
LOCAL H%, DATA%

NEW INDIVIDU, DATA%

; Création du fichier binaire "CLIENTS.CLI" (il est ouvert s'il existe déjà)
MOVE F_CREATE%(SIZEOF INDIVIDU, "CLIENTS.CLI") TO H%
IF F_ERROR%<>0
MESSAGE "Erreur", "Impossible de créer ou d'ouvrir le fichier"
EXIT
ENDIF

; Ecriture de nouvelles données en mémoire
MOVE "Dupont" TO INDIVIDU(DATA%).NOM
MOVE "Jean" TO INDIVIDU(DATA%).PRENOM
; Ecriture de la zone mémoire dans le fichier binaire
F_WRITE H%, DATA%

MOVE "Durand" TO INDIVIDU(DATA%).NOM
MOVE "Pierre" TO INDIVIDU(DATA%).PRENOM
```

```

F_WRITE H%, DATA%

; Fermeture du fichier
F_CLOSE H%
; Libération de la mémoire allouée
DISPOSE DATA%

```

Voir aussi F_CLOSE, F_EOF%, F_ERROR%, F_LOCK, F_OPEN%, F_POS%, F_READ, F_SEEK, F_SIZE%, F_TRUNCATE, F_UNLOCK, F_WRITE, **Fonctions et Instructions T_***, **Language NCL: Instructions LOAD et SAVE**

Fonction F_OPEN% (Librairie NSMisc)

Ouvre un fichier binaire en mode lecture/écriture et retourne son handle.

Syntaxe	F_OPEN% (taille-segment, nom-fichier)		
Paramètres	taille-segment	INTEGER	taille du segment utilisé au sein du fichier
	nom-fichier	CSTRING	nom du fichier à ouvrir
Valeur retournée	INT(4)		

1. Un fichier binaire est un fichier constitué d'une suite d'enregistrements ayant toujours la même structure. Un enregistrement est décrit à l'aide d'un segment NCL. Un fichier binaire n'est pas lisible à l'aide d'un éditeur de texte.
2. La taille du segment peut être spécifiée aisément à l'aide de l'opérateur SIZEOF.
3. Le handle retourné permet ensuite d'accéder au fichier à l'aide des autres fonctions et instructions F_+% de lecture/écriture.
4. Après l'ouverture, le pointeur est positionné sur le premier enregistrement. Pour pouvoir modifier un enregistrement ou écrire après le dernier enregistrement, il faut repositionner le pointeur à l'aide de F_SEEK.
5. Lorsque le fichier est inexistant, F_OPEN% provoque une erreur. Celle-ci doit être lue à l'aide de F_ERROR%.
6. Des parenthèses utilisées au sein de la chaîne nom-fichier permettent d'utiliser les variables d'environnement :

```
; Ouvre le fichier SAMPLE.TXT situé dans le répertoire spécifié par la variable
d'environnement NS-LST
MOVE F_OPEN%(..., "(NS-LST)SAMPLE.TXT") TO H%
```

Exemple :

```
; Déclaration du segment utilisé
SEGMENT INDIVIDU
STRING NOM(31)
STRING PRENOM(31)
ENDSEGMENT

NEW INDIVIDU, DATA%

; Lecture du fichier binaire "CLIENTS.CLI"
MOVE F_OPEN%(SIZEOF INDIVIDU, "CLIENTS.CLI") TO H%
IF F_ERROR%<>0
MESSAGE "Erreur", "Impossible d'ouvrir le fichier"
EXIT
ENDIF

MOVE 0 TO I%

WHILE NOT F_EOF%(H%)
F_READ H%, DATA%
MOVE I%+1 TO I%
MESSAGE "Client numéro" && I%, INDIVIDU(DATA%).NOM && INDIVIDU(DATA%).PRENOM
ENDWHILE

F_CLOSE H%
DISPOSE DATA%
```

Voir aussi [F_ROOPEN%](#), [F_CREATE%](#), [F_READ](#), [F_WRITE](#)

Fonction **F_ROOPEN%** ([Librairie NSMisc](#))

Ouvre un fichier binaire en mode lecture uniquement (read only) et retourne son handle.

Syntaxe	F_ROOPEN% (taille-segment, nom-fichier)		
Paramètres	taille-segment	INTEGER	taille du segment utilisé au sein du fichier
	nom-fichier	CSTRING	nom du fichier à ouvrir
Valeur retournée	INT(4)		

1. Hormis l'ouverture en lecture uniquement, cette fonction est l'équivalent de F_OPEN%.

2. Voir les commentaires sur F_OPEN%.

Voir aussi F_OPEN%, F_CREATE%, F_READ, F_WRITE

Instruction **F_BLOCKREAD** (Librairie NSMisc)

Lit plusieurs enregistrements (ou blocs) à la fois dans un fichier binaire.

Syntaxe	F_BLOCKREAD <i>handle-fichier, adresse-buffer, nb-blocs-buffer, nb-blocs-lus</i>			
Paramètres	handle-fichier	INT(4)	I	handle du fichier à lire
	adresse-buffer	INT(4)	I	adresse du buffer de recopie des enregistrements lus
	nb-blocs-buffer	INT(2)	I	nombre d'enregistrements à lire
	nb-blocs-lus	INT(2)	I/O	nombre d'enregistrements effectivement lus

1. La lecture se fait à partir de la position courante du pointeur. Celui-ci est ensuite positionné après le dernier enregistrement effectivement lu. Les enregistrements lus sont recopiés dans le buffer (ou la zone mémoire) d'adresse adresse-buffer.

2. Le buffer de recopie doit avoir pour taille minimum celle du segment spécifié lors de F_OPEN% ou de F_CREATE%.

3. Le nombre d'enregistrements à lire doit toujours être inférieur ou égal au nombre d'enregistrements pouvant être contenu dans le buffer de recopie.

4. nb-blocs-lus est mis à jour par l'instruction une fois la lecture et la recopie effectuées. Cette variable contient le nombre d'enregistrements effectivement lus dans le fichier.

Exemple :

```
; Déclaration du segment utilisé
; Événement INIT de la fenêtre
SEGMENT INDIVIDU
STRING NOM(31)
```



```

STRING PRENOM(31)
ENDSEGMENT

; Evénement EXECUTED du bouton d'écriture dans le fichier
LOCAL H%, DATA%
; Définition d'une zone de 100 enregistrements
LOCAL INDIVIDU BUFF[100]
LOCAL INT NB_BLOCS_BUFF%(2)
LOCAL INT NB_BLOCS%(2)

; Ouverture du fichier binaire "CLIENTS.CLI"
MOVE F_OPEN%(SIZEOF INDIVIDU, "CLIENTS.CLI") TO H%
IF F_ERROR%<>0
MESSAGE "Erreur", "Impossible d'ouvrir le fichier"
EXIT
ENDIF

; Lecture des 2 premiers enregistrements
F_BLOCKREAD H%,@BUFF, 2, NB_BLOC %
IF F_ERROR% <> 0
MESSAGE "Erreur", "Impossible de lire les 2 premiers enregistrements"
ELSE
MESSAGE "Lecture OK", NB_BLOCS% && "enregistrements ont été lus "
ENDIF

; Fermeture du fichier
F_CLOSE H%

```

Voir aussi [F_READ](#), [F_BLOCKWRITE](#), [F_WRITE](#)

Instruction **F_BLOCKWRITE** (Librairie NSMisc)

Ecrit plusieurs enregistrements (ou blocs) à la fois dans un fichier binaire.

Syntaxe	F_BLOCKWRITE <i>handle-fichier, adresse-buffer, nb-blocs-buffer, nb-blocs-écrits</i>			
Paramètres	handle-fichier	INT(4)	I	handle du fichier à mettre à jour
	adresse-buffer	INT(4)	I	adresse du buffer contenant les enregistrements à écrire.
	nb-blocs-buffer	INT(2)	I	nombre d'enregistrements à écrire
	nb-blocs-écrits	INT(2)	I/O	nombre d'enregistrements effectivement écrits

- 1. Les enregistrements à écrire dans le fichier doivent avoir été préalablement écrits dans le buffer (ou la zone mémoire) d'adresse adresse-buffer.
- 2. Le buffer doit avoir pour taille minimum celle du segment spécifié lors de `F_OPEN%` ou de `F_CREATE%`.
- 3. L'écriture dans le fichier se fait à partir de la position courante du pointeur. Celui-ci est ensuite positionné après le dernier enregistrement écrit.
- 4. Le nombre d'enregistrements à écrire dans le fichier doit correspondre à ceux effectivement écrits dans le buffer. Il est donc toujours inférieur ou égal au nombre d'enregistrements pouvant y être contenu.
- 5. nb-blocs-écrits est mis à jour par l'instruction une fois l'écriture sur le fichier effectuée. Cette variable contient le nombre d'enregistrements effectivement écrits dans le fichier.

Exemple :

```
; Déclaration et ouverture du fichier similaire à ceux de l'exemple F_BLOCKREAD
; Ecriture des enregistrements dans le buffer
BUFF[0].NOM = "Martin"
BUFF[0].PRENOM = "Paul"
BUFF[1].NOM = "Dubois"
BUFF[1].PRENOM = "Marie"

; Positionnement en fin de fichier pour rajouter les enregistrements au fichier
F_SEEK H%,F_SIZE%(H%)
; Ecriture dans le fichier
F_BLOCKWRITE H%,@BUFF, NB_BLOCS_BUFFER%, NB_BLOCS%
IF F_ERROR% <> 0
MESSAGE "Erreur", "Impossible d'ajouter des enregistrements"
ELSE
MESSAGE "Ecriture OK", NB_BLOCS% && "enregistrements ont été écrits"
ENDIF

; Fermeture du fichier
F_CLOSE H%
```

Voir aussi [F_BLOCKREAD](#), [F_READ](#), [F_WRITE](#)

Instruction `F_READ` ([Librairie NSMisc](#))

Lit un enregistrement à la position courante dans un fichier binaire.

Syntaxe	<code>F_READ handle-fichier, adresse-segment</code>		
Paramètres	handle-fichier	INT(4)	handle du fichier
	adresse-segment	INT(4)	adresse du segment

			recevant les données lues dans le fichier
--	--	--	---

1. Le segment dont l'adresse est passée dans adresse-segment doit être identique à celui utilisé lors de la création ou l'ouverture du fichier.
2. F_READ incrémente automatiquement la position courante. Celle-ci correspond donc à l'enregistrement suivant une fois la lecture effectuée.
3. Il est possible de lire plusieurs enregistrements à la fois à l'aide de F_BLOCKREAD.

Exemple :

```
; Le segment utilisé et l'ouverture du fichier sont supposés avoir été effectués comme
décris dans l'exemple illustrant F_OPEN%
; Lecture de tous les enregistrements
WHILE NOT F_EOF%(H%)
F_READ H%, DATA%
MOVE I%+1 TO I%
MESSAGE "Client numéro" && I%, INDIVIDU(DATA%).NOM && INDIVIDU(DATA%).PRENOM
ENDWHILE

F_CLOSE H%
DISPOSE DATA%
```

Voir aussi F_BLOCKREAD, F_POS%, F_SEEK, F_WRITE

Instruction F_WRITE (Librairie NSMisc)

Écrit un segment à la position courante dans un fichier binaire.

Syntaxe	F_WRITE handle-fichier, adresse-segment		
Paramètres	handle-fichier	INT(4)	handle du fichier
	adresse-segment	INT(4)	adresse du segment à écrire

1. Le segment à écrire doit être identique à celui utilisé lors de la création ou l'ouverture du fichier.
2. F_WRITE incrémente automatiquement la position courante. Celle-ci correspond donc à l'enregistrement suivant une fois l'écriture effectuée.

3. Il est possible d'écrire plusieurs enregistrements à la fois à l'aide de F_BLOCKWRITE.

Exemple :

Voir exemples illustant F_CREATE% et F_OPEN%

Voir aussi F_BLOCKWRITE, F_POS%, F_READ, F_SEEK

Instruction F_BYTEREAD (Librairie NSMisc)

F_BYTEREAD lit un certain nombre d'octets dans le fichier à partir de la position courante du pointeur. Le pointeur courant du fichier est ensuite repositionné en fonction du nombre d'octets lus.

Syntaxe	F_BYTEREAD h%, data, nbbytes%, nbread%			
Paramètres	h%	INT(4)	I	Handle d'un fichier précédemment ouvert avec la fonction F_OPEN.
	data%	POINTER	I	Adresse du buffer de destination
	nbbytes%	INTEGER	I	Nombre d'octets à lire.
	nbread%	INTEGER	O	Nombre d'octets réellement lus.

- 1. Le nombre d'octets lu est indépendant de la taille de l'enregistrement précisé à l'appel de la fonction F_OPEN%.
- 2. data% est l'adresse de la zone mémoire dans laquelle est écrit les blocs lors d'une lecture. Cette zone mémoire doit avoir pour taille minimum celle précisée par le paramètre nbbytes%.
- 3. Le nombre d'octets indiqué dans nbbytes% doit toujours être inférieur ou égal à la taille de la zone mémoire data%.
- 4. nbbytes% et nbread% ont une taille de deux octets en mode 16 bits et de quatre octets en mode 32 bits.

Exemple :

```
local integer NB_BYTES_LUS%
local BUFF%, hFile%
hFile% = F_Open% (10,'c:\autoexec.bat')
if MiscError% <> 0
Message 'ERREUR','fichier c:\autoexec.bat non ouvert'
Exit
endif
new 100, BUFF%
```

```
F_Bytread hFile%, BUFF%, 100, NB_BYTES_LUS%
if (MiscError% <> 0) or (NB_BYTES_LUS% <> 100)
Message 'ERREUR','Erreur ' & MiscError% & ' octets lu:' & NB_BYTES_LUS%
endif
dispose BUFF%
F_Close (hFile%)
```

Voir aussi [F_BYTEWRITE](#)

Instruction **F_BYTEWRITE** (Librairie NSMisc)

F_BYTEWRITE écrit un certain nombre d'octets dans le fichier à partir de la position courante du pointeur. Le pointeur courant du fichier est ensuite repositionné en fonction du nombre d'octets écrits.

Syntaxe	F_BYTEWRITE <i>h%, data, nbbytes%, nbread%</i>			
Paramètres	h%	INT(4)	I	Handle d'un fichier précédemment ouvert avec la fonction F_OPEN% .
	data%	pointer	I	Adresse du buffer de destination
	nbbytes%	integer	I	Nombre d'octets à écrire.
	nbread%	integer	O	Nombre d'octets réellement écrits.

1. Le nombre d'octets écrits est indépendant de la taille de l'enregistrement précisé à l'appel de la fonction [F_OPEN%](#).
2. data% est l'adresse de la zone mémoire à partir de laquelle est copié le bloc lors d'une écriture. Cette zone mémoire doit avoir pour taille minimum celle précisée par le paramètre nbbytes%.
3. Le nombre d'octets indiqué dans nbbytes% doit toujours être inférieur ou égal à la taille de la zone mémoire data%.
4. nbbytes% et nbread% ont une taille de deux octets en mode 16 bits et de quatre octets en mode 32 bits.

Exemple :

```
local integer NB_BYTES_LUS%
local BUFF%, hFile%
hFile% = F_Open% (10,'c:\autoexec.bat')
if MiscError% <> 0
Message 'ERREUR','fichier c:\autoexec.bat non ouvert'
Exit
endif
new 100, BUFF%
F_Bytread hFile%, BUFF%, 100, NB_BYTES_LUS%
if (MiscError% <> 0) or (NB_BYTES_LUS% <> 100)
Message 'ERREUR','Erreur ' & MiscError% & ' octets lu:' & NB_BYTES_LUS%
endif
dispose BUFF%
F_Close (hFile%)
```

Voir aussi F_BYTEREAD

Instruction **F_SEEK** (Librairie NSMisc)

Modifie la position courante dans un fichier binaire.

Syntaxe	F_SEEK <i>handle-fichier, position</i>			
Paramètres	handle-fichier	INT(4)	I	handle du fichier
	position	INT(4)	I	nouvelle position courante dans le fichier

La position est le numéro d'ordre des enregistrements dans le fichier, le premier enregistrement ayant pour position 0 (zéro).

Exemple :

```
; Le segment utilisé et l'ouverture du fichier sont supposés avoir été effectués comme
décrit dans l'exemple illustrant F_OPEN%
; Positionnement sur le dernier enregistrement
F_SEEK H%, F_SIZE%(H%)-1
; Positionnement en fin de fichier, pour écrire après le dernier enregistrement
F_SEEK H%, F_SIZE%(H%)
```

Voir aussi F_POS%, F_EOF%

Instruction **F_TRUNCATE** (Librairie NSMisc)

Tronque un fichier binaire à partir de la position courante.

Syntaxe	F_TRUNCATE <i>handle-fichier</i>			
Paramètre	handle-fichier	INT(4)	I	handle du fichier à

			mettre à jour
--	--	--	---------------

Tronquer signifie supprimer les enregistrements de la position courante incluse jusqu'à la fin du fichier. Le fichier ne contient plus que les enregistrements précédant la position courante.

Voir aussi F_CREATE%, F_OPEN%, F_ERROR%, F_POS%, F_READ, F_SEEK, F_SIZE%

Instruction F_BYTESEEK (Librairie NSMisc)

F_BYTESEEK positionne le pointeur courant à la position sélectionnée.

Syntaxe	F_BYTESEEK <i>h%, p%</i>		
Paramètres	<i>h%</i>	INT(4)	Handle d'un fichier précédemment ouvert avec la fonction <u>F_OPEN%</u>
	<i>p%</i>	INT(4)	nouvelle position du pointeur

La nouvelle position du pointeur est indépendante de la taille de l'enregistrement précisé à l'appel de la fonction F_OPEN%.

Instruction F_LOCK (Librairie NSMisc)

Bloque l'enregistrement courant dans un fichier binaire.

Syntaxe	F_LOCK <i>handle-fichier</i>		
Paramètre	handle-fichier	INT(4)	handle du fichier à mettre à jour

Un enregistrement bloqué ne peut plus être modifié. Il redevient modifiable en utilisant F_UNLOCK.

Exemple :

```

; Le segment utilisé et l'ouverture du fichier sont supposés avoir été effectués comme
décrit dans l'exemple illustrant F_OPEN%

; mémorisation de la position courante
MOVE POS%(H%) TO MEMPOS%
; blocage de l'enregistrement courant
F_LOCK H%

; *** traitements
; repositionnement sur l'enregistrement bloqué
F_SEEK H%, MEMPOS%
; Libération de l'enregistrement
F_UNLOCK H%

```

Voir aussi F_ERROR%, F_UNLOCK

Instruction **F_UNLOCK** (Librairie NSMisc)

Libère l'enregistrement courant dans un fichier binaire.

Syntaxe	F_UNLOCK <i>handle-fichier</i>		
Paramètre	handle-fichier	INT(4)	I handle du fichier à mettre à jour

L'enregistrement doit avoir été bloqué par F_LOCK. Une fois libéré il redevient modifiable.

Exemple :

```

; Le segment utilisé et l'ouverture du fichier sont supposés avoir été effectués comme
décrit dans l'exemple illustrant F_OPEN%
; mémorisation de la position courante
MOVE POS%(H%) TO MEMPOS%
; blocage de l'enregistrement courant
F_LOCK H%

; *** traitements
; repositionnement sur l'enregistrement bloqué
F_SEEK H%, MEMPOS%
; Libération de l'enregistrement
F_UNLOCK H%

```

Voir aussi F_ERROR%, F_LOCK

Fonction **F_EOF%** (Librairie NSMisc)

Indique si le pointeur de position courante est situé en fin de fichier (binaire).

Syntaxe	F_EOF% (<i>handle-fichier</i>)
---------	---

Paramètre	handle-fichier	INT(4)	I	handle du fichier
Valeur retournée	INT(1) TRUE% : Le pointeur de position courante est situé en fin de fichier. FALSE% : Le pointeur n'est pas situé en fin de fichier.			

Exemple :

Voir exemples illustrant F_CREATE% et F_OPEN%

Voir aussi F_POS%

Fonction F_ERROR% (Librairie NSMisc)

Retourne le dernier code d'erreur dû à la gestion de fichiers binaires.

Syntaxe	F_ERROR%
Valeur retournée	INT(4) 0 : Aucune erreur non nulle : Erreur dans la dernière fonction ou instruction F_*% utilisée. Les différents codes d'erreur sont donnés dans A propos de Codes d'erreur NSMisc .

Il est conseillé d'appeler F_ERROR% après chaque appel à une fonction ou instruction F_*% de façon à vous assurer que cette dernière s'est bien déroulée.

Exemple :

Voir exemples illustrant F_CREATE% et F_OPEN%

Voir aussi F_CLOSE, F_CREATE%, F_EOF%, F_LOCK, F_OPEN%, F_POS%, F_READ, F_SEEK, F_SIZE%, F_TRUNCATE, F_UNLOCK, F_WRITE, FCOPYFILE

Fonction F_POS% (Librairie NSMisc)

Retourne la valeur du pointeur de position courante dans un fichier binaire.

Syntaxe	F_POS% (handle-fichier)			
Paramètre	handle-fichier	INT(4)	I	handle du fichier
Valeur retournée	INT(4)			

1. La position de l'enregistrement lu ou écrit dans le fichier binaire est mémorisée dans un pointeur. Cette position est le numéro d'ordre des

enregistrements dans le fichier, le premier enregistrement ayant pour position 0 (zéro).

2. La position courante est automatiquement incrémentée après une opération de lecture ou d'écriture, ce qui signifie qu'elle correspond à l'enregistrement qui suit celui qui vient d'être lu ou écrit.

Exemple :

```
; Le segment utilisé et l'ouverture du fichier sont supposés avoir été effectués comme
décrit dans l'exemple illustrant F_OPEN%
MOVE 0 TO I%
; Recherche de Jean CHEVALIER et changement du prénom en Pierre
WHILE NOT F_EOF%(H%)
F_READ H%, DATA%
IF INDIVIDU(DATA%).NOM = "CHEVALIER" AND INDIVIDU(DATA%).PRENOM = "Jean"
; repositionnement du pointeur sur l'enregistrement qui vient d'être lu
F_SEEK H%,F_POS%(H%) -1
MOVE "Pierre" TO INDIVIDU (DATA%).PRENOM
F_WRITE H%,DATA%
BREAK;
ENDIF
ENDWHILE

F_CLOSE H%
DISPOSE DATA%
```

Voir aussi F_EOF%, F_SEEK

Fonction F_SIZE% (Librairie NSMisc)

Retourne la taille d'un fichier binaire.

Syntaxe	F_SIZE% (file-handle)		
Paramètre	file-handle	INT(4)	handle de fichier
Valeur retournée	INT(4)		

La taille retournée est indiquée en nombre d'enregistrements.

Voir aussi F_CREATE%, F_OPEN%

Fonction F_LoadFile\$ (Librairie NSMisc)

Retourne la totalité du contenu d'un fichier, indiqué en paramètre, dans une seule chaîne de caractères de type DynStr. Ainsi, les sauts de ligne sont présents comme caractères de contrôle.

Syntaxe	F_LoadFile\$ FileName\$
---------	-------------------------

Paramètre	Filename\$	CSTRING	I	nom du fichier
Valeur retournée	DYNSTR			

Il est conseillé d'appeler MISCERROR% après un appel à cette fonction de façon à vous assurer que tout s'est bien déroulé.

Exemple :

```
F_LoadFile$ "File02.txt"
IF MISCERROR% = 0
    MESSAGE "OK", "Chargement du fichier effectué !"
ELSE
    MESSAGE "Erreur" && MISCERROR%, "Chargement du fichier non effectué "
ENDIF
```

Voir aussi F_SaveFile

Instruction F_SaveFile (Librairie NSMisc)

Permet de sauvegarder du texte dans un fichier.

Syntaxe	F_SAVEFILE FileName\$, Text\$			
Paramètres	Filename\$	CSTRING	I	nom du fichier
	Text\$	DYNSTR	I	contenu du fichier

Il est conseillé d'appeler MISCERROR% après un appel à cette instruction de façon à vous assurer que tout s'est bien déroulé.

Exemple :

```
F_SaveFile "Fichier1.txt", "L'instruction F_SaveFile permet de \ sauvegarder du texte dans un fichier"
IF MISCERROR% = 0
    MESSAGE "OK", "Sauvegarde effectuée !"
ELSE
    MESSAGE "Erreur" && MISCERROR%, "Sauvegarde non effectuée"
ENDIF
```

Voir aussi F_LoadFile\$

Manipulation de contrôles et fenêtres

Constantes DI_ *% (Librairie NSMisc)

Valeurs indiquant la famille d'un contrôle.

Syntaxe	Déclaration interne	Description
DI_BITMAP%	15	Images Bitmap
DI_CHECKBOX%	5	Contrôle CheckBox
DI_CHECKBOX3%	6	Contrôle CheckBox à trois états (option "3 States" cochée dans sa boîte Info).
DI_COMBBOX%	11	Contrôle ComboBox
DI_COMBBOXE%	12	Contrôle ComboBox with Entry Field
DI_CONTROL%	16	Contrôle Template
DI_CUSTOM%	24	Contrôle Custom (personnalisé)
DI_ENTRYFIELD%	7	Contrôle Entry Field
DI_GROUPBOX%	2	Contrôle GroupBox
DI_HSCROLL%	9	Contrôle Barre horizontale
DI_ICON%	13	Contrôle Icône
DI_LISTBOX%	10	Contrôle ListBox
DI_MENUITEM%	23	Contrôle Menu Item
DI_MLE%	14	Contrôle MLE
DI_PUSHBUTTON%	3	Contrôle PushButton
DI_RADIOBUTTON%	4	Contrôle RadioButton
DI_STATICTEXT%	1	Contrôle Static Text
DI_VSCROLL%	8	Contrôle Barre verticale
DI_TREEBOX%	25	Contrôle TreeBox
DI_WINCMNCTRL%	26	

1. Ces valeurs sont retournées lors de la lecture du paramétrage dynamique .KIND d'un contrôle.

2. Lorsque le paramètre dynamique .KIND est égal à DI_WINCMNCTRL, le sous-type (.SUBKIND) correspond à une des constantes suivantes :

Syntaxe	Déclaration interne
DI_WCC_Animation%	\$40
DI_WCC_DateTimePicker%	\$42

DI_WCC_HotKey%	\$44
DI_WCC_ListView%	\$46
DI_WCC_MonthCalendar%	\$47
DI_WCC_ProgressBar%	\$48
DI_WCC_TrackBar%	\$4A
DI_WCC_TreeView%	\$4B

Exemple 1 :

```
GLOBAL CONTROL CTRL
...
IF CTRL.KIND = DI_ENTRYFIELD%
MESSAGE "Le contrôle CTRL", "est un Entry-Field"
ENDIF
```

Exemple 2 :

```
; Il est impossible de changer la famille d'un contrôle
MOVE DI_PUSHBUTTON% TO ENTRY00001.KIND ; aucun effet !
```

Voir aussi Paramétrage dynamique .KIND

Instruction MAKECONTROL (Librairie NSMisc)

Permet de créer un contrôle en l'initialisant à partir de l'index du contrôle.

Syntaxe	MAKECONTROL ctrl, parentwnd, ctrlid%			
Paramètres	ctrl	CONTROL	I/O	nom du contrôle
	parentwnd	POINTER	I	fenêtre mère
	ctrlid%	INTEGER	I	identifiant du contrôle ID

Exemple :

```
MAKECONTROL C, WindowFromControl%(X), GetControlID%(X)
; est identique à @C = X
```

Instruction FIRSTCONTROL (Librairie NSMisc)

Retourne le premier contrôle d'une fenêtre.

Syntaxe	FIRSTCONTROL <i>handle-fenêtre, var-contrôle</i>			
Paramètres	handle-fenêtre	POINTER	I	handle de la fenêtre
	var-contrôle	CONTROL	I/O	premier contrôle de la fenêtre

1. L'ordre des contrôles est uniquement celui dans lequel ils sont sauvegardés, il n'y a donc aucun lien avec l'apparence physique de la fenêtre ou avec le cheminement du focus. Par conséquent, il est recommandé de ne pas faire de supposition quant à l'ordre des contrôles. Les instructions FIRSTCONTROL et NEXTCONTROL ne servent qu'à obtenir tous les contrôles figurant dans la fenêtre.
2. Le contrôle suivant est obtenu à l'aide de NEXTCONTROL.

Exemple :

```
LOCAL CONTROL CTRL
```

```
FIRSTCONTROL SELF%, CTRL
```

```
MESSAGE "Le premier contrôle trouvé est situé en", CTRL.X && CTRL.Y
```

Voir aussi NEXTCONTROL, CONTROLNAME\$, WINDOWFROMCONTROL%

Instruction NEXTCONTROL (Librairie NSMisc)

Fournit le contrôle suivant de la recherche démarrée par les précédents FIRSTCONTROL et NEXTCONTROL.

Syntaxe	NEXTCONTROL <i>var-contrôle</i>			
Paramètre	var-contrôle	CONTROL	I/O	contrôle suivant de la fenêtre

1. Lorsqu'il n'y a plus de contrôle dans la fenêtre, la fonction MISCERROR% retourne un code d'erreur différent de zéro.
2. Il n'y a aucun lien entre l'ordre des contrôles trouvés par NEXTCONTROL et l'apparence physique de la fenêtre. Aucun non plus avec le cheminement du focus.

3. Il est impératif que la variable contrôle de chaque NEXTCONTROL soit la même que celle utilisée par FIRSTCONTROL car le contrôle qui y figure sert de référence pour obtenir le suivant.

Exemple :

```
; Affichage des noms et blocage de tous les contrôles d'une fenêtre.
LOCAL CONTROL TEMPCTRL

FIRSTCONTROL SELF%, TEMPCTRL

WHILE MISCERROR% = 0
  MESSAGE "Control name" , CONTROLNAME$(TEMPCTRL)
  LOCK TEMPCTRL
  NEXTCONTROL TEMPCTRL
ENDWHILE

; déblocage d'un bouton, Il faut pouvoir fermer la fenêtre
UNLOCK PB_CANCEL
```

Voir aussi FIRSTCONTROL, CONTROLNAME\$, WINDOWFROMCONTROL%

Fonction CONTROLNAME\$ (Librairie NSMisc)

Retourne le nom d'un contrôle.

Syntaxe	CONTROLNAME\$ (contrôle)			
Paramètre	contrôle	CONTROL		contrôle
Valeur retournée	CSTRING			

Le nom d'un contrôle est le contenu du champ Name de sa boîte Info.

Pour que CONTROLNAME\$ fonctionne en généré, il faut IMPERATIVEMENT utiliser l'option /NAMES avec le module de génération, ou avoir coché l'option "Symbols Info" dans la boîte de génération

Exemple 1 :

```
LOCAL CONTROL CTRL
FIRSTCONTROL SELF%, CTRL
MESSAGE "Nom du premier contrôle :", CONTROLNAME$(CTRL)
```

Exemple 2 :

```
LOCAL CONTROL CTRL
MOVE ENTRY00001 TO @CTRL
MESSAGE "Nom du contrôle", CONTROLNAME$(CTRL)
; affiche ENTRY00001
```

Voir aussi WINDOWFROMCONTROL%, WINDOWNAME\$

Fonction WINDOWNAME\$ (Librairie NSMisc)

Retourne le nom d'une fenêtre.

Syntaxe	WINDOWNAME\$ (window-handle)		
Paramètre	window-handle	POINTER	handle de la fenêtre
Valeur retournée	CSTRING		

1. Le nom d'une fenêtre est le contenu du champ "Name" de la boîte Window Info et qui est utilisé pour appeler la fenêtre lors d'un CALL ou OPEN..
2. Pour que WINDOWNAME\$ fonctionne en généré, il faut IMPERATIVEMENT utiliser l'option /NAMES avec le module de génération, ou avoir coché l'option "Symbols Info" dans la boîte de génération.

Exemple 1:

```
MESSAGE "Le nom de la fenêtre courante est :", WINDOWNAME$(SELF%)
```

Exemple 2 :

```
MESSAGE "Le nom de la fenêtre principale est :", WINDOWNAME$(MAINWINDOW%)
```

Voir aussi CONTROLNAME\$, WINDOWFROMCONTROL%

Fonction SELECTBYVALUE% (Librairie NSMisc)

Permet de rechercher du texte ou de changer l'état de sélection d'une ou de plusieurs lignes d'un contrôle.

Syntaxe	SELECTBYVALUE% (ctl, mode%, start%, text\$)		
Paramètres	ctl	CONTROL	nom du contrôle
	mode%	INTEGER	combinaison de constantes <u>SBVM *%</u>
	start%	INTEGER	indique la première ligne à partir de laquelle

			la recherche commence
	text\$	CSTRING	texte recherché à mettre entre guillemets
Valeur retournée	INTEGER		

1. Pour pouvoir utiliser cette fonction sur un Custom Control ou un template, celui-ci doit convenablement implémenter les événements SELECTED, LINECOUNT et GETVALUE (en particulier le cas où (PARAM2% = 0) AND (PARAM1% <> DEFRET%) pour ce dernier).
2. Les actions possibles sont (une seule action possible à la fois) :
 - a) SBVM_FIND% retourne l'index du premier élément trouvé.
 - b) SBVM_SEL% sélectionne les lignes qui correspondent.
 - c) SBVM_UNSEL% désélectionne les lignes qui correspondent.
 - d) SBVM_CHANGE% inverse la sélection des lignes qui correspondent.
3. Les options de comparaison sont (les trois dernières options s'excluent) :
 - a) SBVM_NOCASE% considère que les minuscules et les majuscules sont identiques.
 - b) SBVM_PART% la chaîne recherchée doit être présente dans la ligne.
 - c) SBVM_EXACT% la chaîne recherchée doit être identique à la ligne.
 - d) SBVM_REGEXP% la chaîne recherchée est traitée avec COMPILEREGREXPR% et comparée aux lignes du contrôle avec EXECUTEREGREXPR%.
4. L'option SBVM_ONE% indique une action sur la première ligne correspondante alors que SBVM_ALL% (interdite avec SBVM_FIND%) indique qu'il faut appliquer l'action à toutes les lignes correspondantes.
5. L'option SBVM_FWD% indique une recherche ascendante (numéros de lignes croissants) alors que SBVM_BACK% indique une recherche descendante (numéros de lignes décroissants).
6. Pour les List Box et Combo Box, on peut aussi ajouter les options suivantes:
 - a) SBVM_STRPPA% ignore le texte des attributs de présentation (" {...}" en début de colonne).
 - b) SBVM_STRPBM% ignore les handles de bitmap et d'icône.
 - c) SBVM_STRP0W% ignore le texte et les attributs de présentation des colonnes de largeur 0.

d) SBVM_STRPTP% ignore les pointeurs sur du texte (attribut de présentation "{P[Y]}").

e) SBVM_EXPTP% tient compte du texte référencé par pointeur comme s'il était présent à la place du pointeur.

Voir aussi Constantes SBVM *, GETCONTROLTEMPLATE, GETCONTROLID%, MAKECONTROL

Constantes SBVM_*

Indiquent les options de recherche et l'action à effectuer sur la ou les lignes correspondantes.

Syntaxe	Déclaration interne	Description
SBVM_FIND%	\$0000	retourne l'index du premier élément trouvé.
SBVM_SEL%	\$0001	sélectionne les lignes qui correspondent.
SBVM_UNSEL%	\$0002	désélectionne les lignes qui correspondent.
SBVM_CHANGE%	\$0003	inverse la sélection des lignes qui correspondent.
SBVM_M_ACT%	\$0003	
SBVM_NOCASE%	\$0004	considère que les minuscules et les majuscules sont identiques.
SBVM_PART%	\$0000	la chaîne recherchée doit être présente dans la ligne.
SBVM_EXACT%	\$0008	la chaîne recherchée doit être identique à la ligne.
SBVM_REGEX%	\$0010	la chaîne recherchée est traitée avec COMPILEREGREXPR% et comparée aux lignes du contrôle avec EXECUTEREGREXPR%.
SBVM_M_CMP%	\$0018	
SBVM_ONE%	\$0000	indique une action sur la première ligne correspondante
SBVM_ALL%	\$0020	(interdite avec SBVM_FIND%) indique qu'il faut appliquer l'action à toutes les lignes correspondantes.
SBVM_M_RNG%	\$0020	

SBVM_FWD%	\$0000	indique une recherche ascendante (numéros de lignes croissants)
SBVM_BACK%	\$0040	indique une recherche descendante (numéros de lignes décroissants).
SBVM_M_DIR%	\$0040	
SBVM_STRPPA%	\$0080	ignore le texte des attributs de présentation (" {...}" en début de colonne).
SBVM_STRPBM%	\$0100	ignore les handles de bitmap et d'icône.
SBVM_STRPOW%	\$0200	ignore le texte et les attributs de présentation des colonnes de largeur 0.
SBVM_STRPTP%	\$0400	ignore les pointeurs sur du texte (attribut de présentation "{P[Y]}").
SBVM_EXPTP%	\$0800	tient compte du texte référencé par pointeur comme s'il était présent à la place du pointeur.
SBVM_M_LBO%	\$0F80	

Les constantes SBVM_STRP*% et SBVM_EXPTP% ne sont utilisables qu'avec des List Box ou des Combo Box.

Voir [SELECTBYVALUE](#)

Fonction NSMEMPOS% (Librairie NSMisc)

Retourne la position d'une chaîne dans une autre chaîne.

Syntaxe	NSMEMPOS% (pMem1, pMem2, size%, size2%, last%)		
Paramètres	pMem1	POINTER	pointeur sur le bloc de mémoire cherché
	pMem2	POINTER	pointeur sur le bloc de mémoire dans lequel on

			effectue la recherche
	size%	INTEGER	nombre d'octets du bloc de mémoire cherché
	size2%	INTEGER	nombre d'octets du bloc de mémoire dans lequel on effectue la recherche
	last%	INTEGER	direction de la recherche
Valeur retournée	INTEGER		

1. Indiquer dans le paramètre last% la valeur FALSE% pour effectuer la recherche à partir du début et TRUE% pour effectuer la recherche à partir de la fin.
2. La valeur retournée est l'offset (à partir du début du second bloc), où commence la séquence d'octets recherchée ou -1 si la recherche échoue.
3. La fonction NSMEMPOS% a la même fonctionnalité que la fonction POS% mais uniquement pour des blocs d'octets qui ne se terminent pas par 0.

Voir aussi *POS% (NCL)*, *NSMEMCOMP%*

Instruction **SHORTENPATHNAME** (Librairie NSMisc)

Remplace un nom de fichier long passé en paramètre modifiable par son équivalent court ("Mon document test.txt" devient par exemple "MONDOC~1.TXT").

Syntaxe	SHORTENPATHNAME <i>pathname\$</i>
----------------	--

Paramètre	pathname\$	CSTRING	I/O	nom du fichier
------------------	------------	---------	-----	----------------

Si le nom de fichier comporte un chemin, les noms des répertoires du chemin sont aussi remplacés par les noms courts équivalents.

Fonction WINDOWFROMCONTROL% (Librairie NSMisc)

Retourne le handle de la fenêtre auquel appartient un contrôle donné.

Syntaxe	WINDOWFROMCONTROL% (contrôle)			
Paramètre	contrôle	CONTROL	I	contrôle
Valeur retournée	POINTER			

Une variable de type CONTROL est stockée sur 6 octets, dont 4 contiennent le handle de la fenêtre à laquelle appartient le contrôle. WINDOWFROMCONTROL% retourne ces 4 octets.

Exemple :

```
GLOBAL CONTROL CTRL
; Soit une fenêtre SCREEN, de handle H% et ayant un contrôle ENTRY1
MOVE SCREEN(H%).ENTRY1 TO @CTRL

; WINDOWFROMCONTROL% permet de retrouver H%
MESSAGE "Handle de la fenêtre contenant CTRL", WINDOWFROMCONTROL%(CTRL)

; WINDOWNAME$ permet de retrouver le nom de la fenêtre
MESSAGE "Nom de la fenêtre contenant CTRL", WINDOWNAME$(WINDOWFROMCONTROL%(CTRL))

; CONTROLNAME$ permet de retrouver ENTRY1
MESSAGE "Nom du contrôle stocké dans CTRL", CONTROLNAME$(CTRL)
```

Voir aussi CONTROLNAME\$, WINDOWNAME\$

Fonction GETCONTROLID% (Librairie NSMisc)

La fonction GETCONTROLID% retourne le numéro ou index du contrôle passé en paramètre.

Syntaxe	GETCONTROLID% (ctrl)			
Paramètre	ctrl	CONTROL	I	nom du contrôle
Valeur retournée	INTEGER			

Cette fonction complète la fonction WINDOWFROMCONTROL%. La fonction WINDOWFROMCONTROL% retourne le handle de la fenêtre contenant le contrôle passé en paramètre.

Exemple :

```
; PB04 = nom du contrôle Push Button
MESSAGE "Numéro du contrôle Push Button", GETCONTROLID%(PB04)
; retourne 102
; EF01 = nom du contrôle Entry-Field
MESSAGE "Numéro du contrôle Entry Field", GETCONTROLID%(EF01)
; retourne 101
```

Fonction NSGETMENUWINDOW (Librairie NSMisc)

Retourne le handle de la fenêtre contenant le menu.

Syntaxe	NSGETMENUWINDOW (Wnd)		
Paramètre	wnd	POINTER	pointeur de la fenêtre (self%)
Valeur retournée	POINTER		

1. Si c'est une fenêtre fille MDI, NSGETMENUWINDOW récupère le handle de la fenêtre mère MDI.
2. Permet le partage d'une seule Image List attachée à la fenêtre mère MDI pour des fenêtres filles MDI. Exemple : WIL GETCOUNT%(NSGetMenuWindow(Self%), ...). où Self% est le handle d'une fenêtre fille MDI ou d'une fenêtre non MDI.

Voir aussi WIL DELETEPICTURES, WIL DESTROY, WIL DRAWPICTURE, WIL FREEHASH, WIL GETSIZE, WIL SETCOUNT, WIL APPENDFROMBMP% , WIL FINDORADDONEPICTURE%, WIL FINDPICTURE%, WIL GETCOUNT%, WIL GETPICTURESCOUNT%

Fonction SELFFROMCONTROL% (Librairie NSMisc)

Retourne le paramètre Self% d'un contrôle.

Syntaxe	SELFFROMCONTROL% (ctrl)		
Paramètre	ctrl	CONTROL	nom du contrôle
Valeur retournée	POINTER		

Permet d'appliquer certaines fonctions prenant un handle de fenêtre à un contrôle.

Fonction ISVALIDSELFHANDLE% (Librairie NSMisc)

Cette fonction sert à déterminer si le nombre passé en paramètre est un handle de fenêtre valide.

Syntaxe	ISVALIDSELFHANDLE% (<i>HandleFen%</i>)		
Paramètre	HandleFen%	INT(4)	I Handle de la fenêtre
Valeur retournée	INT(1) Booléen True% ou False%		

1. Il est à noter que le nombre passé en paramètre est un handle au sens NatStar (ou NS-DK), et non pas un handle système. Les handles NatStar (ou NS-DK) sont par exemple ceux qui sont retournés par la fonction SELF%.
2. Cette fonction peut par exemple servir à vérifier si une fenêtre ouverte précédemment par l'application est encore affichée à l'écran ou si elle a été fermée par l'utilisateur.

Exemple :

```
IF ISVALIDSELFHANDLE%(HANDLE%)
...
ENDIF
```

Instruction FOCUSCONTROL (Librairie NSMisc)

Retourne le contrôle ayant le focus dans une fenêtre.

Syntaxe	FOCUSCONTROL <i>handle-fenêtre, variable-contrôle</i>			
Paramètres	handle-fenêtre	POINTER	I	handle de la fenêtre
	variable-contrôle	CONTROL	I/O	contrôle ayant le focus dans la fenêtre

En cas d'erreur, la fonction MISCERROR% renvoie ERROR_INVALID_HANDLE% ou ERROR_NO_MORE_CONTROL%.

Exemple :

```
LOCAL CONTROL CTRL_FOCUS
; Modification de la couleur de fond du contrôle ayant le focus
FOCUSCONTROL SELF%, CTRL_FOCUS
MOVE COL_RED% TO CTRL_FOCUS.BACKCOLOR
```

Voir aussi [GETTABCONTROL](#), [SETTABCONTROL](#)

Instruction GETTABCONTROL (Librairie NSMisc)

Indique quel contrôle d'une fenêtre de classe Dialog doit obtenir le focus lors de l'appui sur la touche [Tab] lorsqu'un contrôle donné a le focus.

Syntaxe	GETTABCONTROL (ctrl-source, ctrl-cible)			
Paramètres	ctrl-source	CONTROL	I	contrôle d'où part le focus
	ctrl-cible	CONTROL	I/O	contrôle où arrive le focus

Le contrôle indiqué est celui qui a été associé statiquement au champ Tab de la boîte Info du contrôle ctrl-source ou qui a été spécifié dynamiquement à l'aide de [SETTABCONTROL](#).

Exemple :

```
local CONTROL CTRLSRC
local CONTROL CTRLCIBLE
; Détermine quel contrôle a le focus
FOCUSCONTROL SELF%, CTRLSRC
; Obtient le nom du contrôle devant obtenir le focus
GETTABCONTROL CTRLSRC, CTRLCIBLE
MESSAGE "GetTabControl", CONTROLNAME$(CTRLCIBLE) && "va obtenir le focus après" &&
CONTROLNAME$(CTRLSRC)
; Cet exemple peut figurer dans l'événement EXECUTED d'un push button de test si ce
dernier a l'option No Focus cochée dans sa boîte Info.
```

Voir aussi [SETTABCONTROL](#), [FOCUSCONTROL](#)

Instruction SETTABCONTROL (Librairie NSMisc)

Spécifie le contrôle d'une fenêtre de classe Dialog qui doit obtenir le focus lors de l'appui sur la touche [Tab] lorsqu'un contrôle donné a le focus.

Syntaxe	SETTABCONTROL (ctrl-source, ctrl-cible)			
Paramètres	ctrl-source	CONTROL	I	contrôle d'où part le focus
	ctrl-cible	CONTROL	I	contrôle où

			arrive le focus
--	--	--	-----------------

Cette instruction est l'équivalent dynamique du champ Tab de la boîte Info du contrôle ctrl-source.

Exemple :

```
; Dans la fenêtre de saisie, le champ "Nom de jeune fille" obtient le focus lors de
l'appui sur Tab dans le champ "Nom" si la personne est mariée et de sexe féminin
IF CK_MARRIE% = CHECKED%
IF RB_SEXE = 2
SETTABCONTROL EF_NOM, EF_NOMJEUNEFILLE
SETTABCONTROL EF_NOMJEUNEFILLE, EF_PRENOM
ELSE
SETTABCONTROL EF_NOM, EF_PRENOM
ENDIF
ENDIF
```

Voir aussi [GETTABCONTROL](#), [FOCUSCONTROL](#)

Fonction SETFOCUSCONTROLATTRIBUTES% (Librairie NSMisc)

Détermine les attributs à mettre au point sur un contrôle prenant le focus. Cette fonction est surtout utile pour une icône et pour des applications utilisant des caractères Unicode complexes (idéogrammes, ...) sans refaire les boîtes de dialogue.

Syntaxe	SETFOCUSCONTROLATTRIBUTES% (<i>focusctrlattrs</i>)			
Paramètre	focusctrlattrs	SEGMENT	I/O	segment FOCUSCTRLATTRS
Valeur retournée	INT(1)			

1. Le segment FOCUSCTRLATTRS est défini dans le fichier NSMISC.NCL :

```
SEGMENT FOCUSCTRLATTRS
INT Version ; SIZEOF FOCUSCTRLATTRS (pour version du segment)
INT KindMask ; une constante CKM_*%
CSTRING Font ; nom de la police du contrôle prenant le focus
INT FontSize(2) ; taille de la police du contrôle prenant le focus
INT FontSels(2) ; type de la police (gras,italique,...)
INT MinWidth(2) ; largeur minimale en pixels
INT MinHeight(2) ; hauteur minimale en pixels (pour contrôle MLE uniquement)
ENDSEGMENT
```

2. Les constantes CKM_*% pouvant être définies dans le champ KindMask déterminent le type du contrôle.
3. Cette fonction peut être appelée plusieurs fois avec des valeurs différentes pour le champ KindMask pour fixer des attributs spécifiques à différents types de contrôles.

Exemple :

```
Local FOCUSCTRLATTRS theSeg
theSeg.Version = sizeof FOCUSCTRLATTRS
theSeg.KindMask = CKM_DEF% ; use any of the CKM_* constants
theSeg.Font="Simplified Arabic"
theSeg.FontSize = 14
;GFS_ITALIC% 1
;GFS_UNDERSCORE% 2 GFS_STRIKEOUT% 4 GFS_BOLD% 8
;GFS_OUTLINE% 16
theSeg.FontSels= GFS_OUTLINE%; use any of the GFS_* constants
;bold/italic/... of the focus font
theSeg.MinWidth=14 ; minimum width in pixels
theSeg.MinHeight= 14 ; minimum height in pixels (for MLE only)
; since we are from 2.61 there is no need to define a return variable
; for the function nor brackets for the parameters
SETFOCUSCONTROLATTRIBUTES% theSeg
```

Voir aussi Constantes CKM_*%

Constantes CKM_*% (Librairie NSMisc)

Détermine le type du contrôle prenant le focus.

Syntaxe	Déclaration interne	Description
CKM_MLE%	\$0001	contrôle MLE.
CKM_EF%	\$0002	contrôle Entry Field.
CKM_FEF%	\$0004	contrôle Filtered Entry Field (pas de caractères >= 255).
CKM_CBE%	\$0008	contrôle Combo Box with Entry field.
CKM_FCBE%	\$0010	contrôle Filtered Combo Box with Entry field (pas de caractères >= 255).
CKM_CB%	\$0020	contrôle Combo Box.
CKM_PB%	\$0040	contrôle Push Button
CKM_CK%	\$0080	contrôle Check box
CKM_RB%	\$0100	contrôle Radio Button
CKM_LB%	\$0200	contrôle List Box.
CKM_DEF%	\$000B	contrôle MLE+EF+CBE

Voir aussi SETFOCUSCONTROLATTRIBUTES%

Fonction GETCHECKMODE% (Librairie NSMisc)

Retourne l'état du mode permettant d'inhiber la perte de focus d'un contrôle.

Syntaxe	GETCHECKMODE% (<i>handle-fenêtre</i>)		
Paramètre	handle-fenêtre	POINTER	handle de la fenêtre
Valeur retournée	INT(1) TRUE% : Le mode est activé FALSE% : Le mode n'est pas activé		

Exemple :

```
; Inversion du "Check Mode" pour la fenêtre courante
IF GETCHECKMODE%(SELF%)
SETCHECKMODE SELF%, FALSE%
ELSE
SETCHECKMODE SELF%, TRUE%
ENDIF
```

Voir aussi SETCHECKMODE

Instruction SETCHECKMODE (Librairie NSMisc)

Active ou désactive le "Check Mode" permettant d'empêcher la perte de focus de tout contrôle appartenant à une fenêtre.

Syntaxe	SETCHECKMODE <i>handle-fenêtre, mode</i>		
Paramètres	handle-fenêtre	POINTER	handle de la fenêtre
	mode	INT(1)	active/désactive le check mode

1. L'activation du "Check Mode" est effectuée lorsque mode vaut TRUE%. Le contrôle courant reçoit alors l'événement CHECK à la place de l'événement LOSEFOCUS lorsqu'il perd le focus. Un RETURN 1 dans le code de l'événement CHECK empêche la perte de focus pour le contrôle.
2. La désactivation du "Check Mode" est effectuée lorsque mode vaut FALSE%. L'événement LOSEFOCUS est alors reçu par tout contrôle perdant le focus.
3. Un Push Button en 'No Checking' n'envoie pas l'événement CHECK même lors d'un changement de focus. De même pour le Push Button associé à la touche [Esc].

Exemple :

```
Événeme : :nt INIT de la fenêtre courante
; Activation du "Check Mode"
SETCHECKMODE SELF%, TRUE%
```

```

Événement CHECK de l'entry field EF_NOM
IF PARAM1% = CHK_LOSEFOCUS%
; perte de focus en cours
IF EF_NOM = ""
BEEP
MESSAGE "Erreur" , "Le champ Nom est obligatoire"
; empêche la perte de focus
RETURN 1
ELSE
RETURN 0
ENDIF
ENDIF

```

Voir aussi GETCHECKMODE%, constantes CHK_*

Constantes CHK_*% (Librairie NSMisc)

Valeurs indiquant la cause de l'événement CHECK.

Syntaxe	Déclaration interne	Description
CHK_LOSEFOCUS%	0	perte de focus du contrôle courant.
CHK_ENDSESSION%	1	arrêt de Windows (ne peut être reçu que sous Windows)
CHK_PUSHBUTTON%	2	appui sur un Push Button.

Voir aussi SETCHECKMODE, événement CHECK

Instruction GETESCAPECONTROL (Librairie NSMisc)

Indique quel contrôle d'une fenêtre de classe Dialog est activé par l'appui sur la touche [Echap].

Syntaxe	GETESCAPECONTROL (handle-fenêtre, contrôle)			
Paramètres	handle-fenêtre	POINTER	I	handle de la fenêtre de classe Dialog
	contrôle	CONTROL	I/O	contrôle activé

Le contrôle indiqué est celui qui a été associé statiquement à la touche [Echap] dans la boîte Info de la fenêtre ou dynamiquement à l'aide de SETESCAPECONTROL. En règle générale, le contrôle ainsi associé est un Push button qui reçoit l'événement EXECUTED lors de l'appui sur [Echap].

Exemple :

```

LOCAL CONTROL CTRL
GETESCAPECONTROL SELF%, CTRL
IF CTRL = PB_CANCEL
MESSAGE "Info", "La touche [Echap] est associée au bouton Cancel"
ENDIF

```

Voir aussi [SETESCAPECONTROL](#), [GETRETURNCONTROL](#), [SETRETURNCONTROL](#)

Instruction **GETRETURNCONTROL** ([Librairie NSMisc](#))

Indique quel contrôle d'une fenêtre de classe Dialog est activé par l'appui sur la touche [Entrée].

Syntaxe	GETRETURNCONTROL (<i>handle-fenêtre, contrôle</i>)			
Paramètres	handle-fenêtre	POINTER	I	handle de la fenêtre de classe Dialog
	contrôle	CONTROL	I/O	contrôle activé

Le contrôle indiqué est celui qui a été associé statiquement à la touche [Entrée] dans la boîte Info de la fenêtre ou dynamiquement à l'aide de [SETRETURNCONTROL](#). En règle générale, le contrôle ainsi associé est un Push button qui reçoit l'événement EXECUTED lors de l'appui sur [Entrée].

Exemple :

```
LOCAL CONTROL CTRL
GETRETURNCONTROL SELF%, CTRL
IF CTRL = PB_OK
MESSAGE "Info", "La touche [Entrée] est associée au bouton OK"
ENDIF
```

Voir aussi [SETRETURNCONTROL](#), [GETESCAPECONTROL](#), [SETESCAPECONTROL](#)

Instruction **SETESCAPECONTROL** ([Librairie NSMisc](#))

Modifie dynamiquement le contrôle d'une fenêtre de classe Dialog devant être activé par l'appui sur la touche [Echap].

Syntaxe	SETESCAPECONTROL (<i>handle-fenêtre, contrôle</i>)			
Paramètres	handle-fenêtre	POINTER	I	handle de la fenêtre de classe Dialog
	contrôle	CONTROL	I	contrôle devant être activé

En règle générale, le contrôle ainsi associé est un Push button qui reçoit l'événement EXECUTED lors de l'appui sur [Echap].

Exemple :

```
IF ISDISABLED% (PB_CANCEL)
SETESCAPECONTROL SELF%,PB_EXIT
ELSE
SETESCAPECONTROL SELF%, PB_CANCEL
ENDIF
```

Voir aussi [GETESCAPECONTROL](#), [GETRETURNCONTROL](#), [SETRETURNCONTROL](#)

Instruction SETRETURNCONTROL ([Librairie NSMisc](#))

Modifie dynamiquement le contrôle d'une fenêtre de classe Dialog devant être activé par l'appui sur la touche [Entrée].

Syntaxe	SETRETURNCONTROL (<i>handle-fenêtre, contrôle</i>)		
Paramètres	handle-fenêtre	POINTER	handle de la fenêtre de classe Dialog
	contrôle	CONTROL	contrôle devant être activé

En règle générale, le contrôle ainsi associé est un Push button qui reçoit l'événement EXECUTED lors de l'appui sur [Entrée].

Exemple :

```
IF ISDISABLED% (PB_OK)
SETRETURNCONTROL SELF%,PB_HELP
ELSE
SETRETURNCONTROL SELF%, PB_OK
ENDIF
```

Voir aussi [GETRETURNCONTROL](#), [GETESCAPECONTROL](#), [SETESCAPECONTROL](#)

Fonction GETWNDCAPTURE ([Librairie NSMisc](#))

Cette fonction retourne le handle de la fenêtre qui a capturé la souris. Ce handle n'est pas du type Natstar (ou NS-DK) mais est un handle connu par le système d'exploitation.

Syntaxe	GETWNDCAPTURE
Valeur retournée	POINTER handle de la fenêtre ayant capturé la souris.

Ce handle peut être transformé en handle NatStar (ou NS-DK) via la fonction GETSELFFROMHWND% de la librairie NSWIN.NCL sous Windows.

Voir aussi GETSELFFROMHWND% de la librairie NSWIN

Constantes ADJUST_*% (Librairie NSMisc)

Les constantes ADJUST_*% sont utilisées conjointement avec le segment SEG_SIZEORPOS et l'événement ADJUSTSIZEORPOS. Cet événement informe une fenêtre que l'utilisateur désire modifier sa position ou sa taille et permet à celle-ci de changer par programme les valeurs des nouvelles positions et tailles.

Syntaxe	Déclaration interne	Description
ADJUST_MOVE%	8	Indique que l'utilisateur désire modifier la position de la fenêtre.
ADJUST_SIZE%	16	Indique que l'utilisateur désire modifier la taille de la fenêtre.

Voir aussi SEG_SIZEORPOS

Segment SEG_SIZEORPOS (Librairie NSMisc)

Ce segment est utilisé conjointement avec les constantes ADJUST_*% et l'événement ADJUSTSIZEORPOS. Cet événement informe une fenêtre que l'utilisateur désire modifier sa position ou sa taille et permet à celle-ci de changer par programme les valeurs des nouvelles position et taille.

Syntaxe	SEGMENT SEG_SIZEORPOS posX posY Width Height ENDSEGMENT	
Champs	posX	INT(4)
	posY	INT(4)
	Width	INT(4)

	Height	INT(4)
--	--------	--------

Voir aussi ADJUST *%

Fonction ISRETORESCCONTROL% (Librairie NSMisc)

Retourne une combinaison des constantes ISRETURNCONTROL%=1 et ISESCAPECONTROL%=2 indiquant si le contrôle passé en paramètre est associé à la touche Return et/ou Escape (le résultat est donc entre 0 et 3).

Syntaxe	ISRETORESCCONTROL% (ctrl)		
Paramètre	ctrl	CONTROL	I nom du contrôle
Valeur retournée	INTEGER		

Voir aussi Constantes IS*%

Constantes IS*% (Librairie NSMisc)

Valeurs indiquant si le contrôle passé en paramètre dans la fonction ISRETORESCCONTROL% est associé à la touche Return et/ou Escape.

Syntaxe	Déclaration interne	Description
ISRETURNCONTROL%	1	contrôle associé à la touche Return
ISESCAPECONTROL%	2	contrôle associé à la touche Escape

Voir aussi ISRETORESCCONTROL%

MDI Window Management

Fonction GETMDISTR\$ (Librairie NSMisc)

Retourne l'intitulé d'une option du menu système ou du menu de positionnement (Chevauchement, Juxtaposition) des fenêtres MDI.

Syntaxe	GETMDISTR\$ (option-menu)		
Paramètre	option-menu	INT(1)	I option de menu à prendre

			en compte
Valeur retournée	CSTRING		

1. L'option de menu à prendre en compte est spécifiée à l'aide d'une constante `MDI_*`.
2. Si, à l'affichage du menu, l'intitulé contient un caractère accélérateur clavier, ce dernier est précédé du caractère tilde (~) dans la chaîne retournée par `GETMDISTR$`.

Exemple :

```
; Lecture de l'intitulé de juxtaposition
LOCAL A$(40)
MOVE GETMDISTR$(MDI_TILE%) TO A$
```

Voir aussi `SETMDISTR`, `MDI_*`

Constantes `MDI_*` (Librairie NSMisc)

Identification d'une option de menu système ou de menu de positionnement des fenêtres MDI.

Syntaxe	Déclaration interne	Description
<code>MDI_CANCEL%</code>	12	Option Cancel (Annulation)
<code>MDI_CASCADE%</code>	8	Option Cascade (Chevauchement)
<code>MDI_CLOSE%</code>	7	Option Close (Arrêt/Fermeture)
<code>MDI_MAXIMIZE%</code>	6	Option Maximize (Agrandissement)
<code>MDI_MINIMIZE%</code>	5	Option Minimize (Réduction)
<code>MDI_MORE%</code>	10	Option More (A suivre)
<code>MDI_MOVE%</code>	3	Option Move (Déplacement)
<code>MDI_NEXT%</code>	2	Option Next (Fenêtre suivante)
<code>MDI_RESTORE%</code>	1	Option Restore (Restauration)
<code>MDI_SELECT%</code>	11	Option Select (Visualisation)
<code>MDI_SIZE%</code>	4	Option Size (Dimensionnement)

MDI_TILE%	9	Option Tile (Juxtaposition)
-----------	---	-----------------------------

1. Les intitulés des options représentées par ces constantes sont lus ou modifiés à l'aide de GETMDISTR\$ et SETMDISTR.
2. L'option correspondant à MDI_MORE% apparaît dans le menu à la suite des noms des fenêtres MDI ouvertes lorsque le nombre de celles-ci est supérieur à neuf. MDI_SELECT% et MDI_CANCEL% correspondent aux intitulés des boutons de la boîte de dialogue affichée suite à la sélection de l'option MDI_MORE%.
3. La modification des intitulés du menu système n'est pas possible. L'ensemble des constantes MDI_*% associées au menu système n'a donc aucun effet lors de leur utilisation avec SETMDISTR.

Voir aussi GETMDISTR\$, SETMDISTR

Instruction SETMDISTR (Librairie NSMisc)

Modifie l'intitulé d'une option du menu système ou du menu de positionnement (Chevauchement, Juxtaposition) des fenêtres MDI.

Syntaxe	SETMDISTR option-menu, chaîne-intitulé		
Paramètres	option-menu	INT(1)	option de menu à modifier
	chaîne-intitulé	CSTRING	nouvel intitulé

1. L'option de menu concernée est précisée sous forme de constante MDI_*%.
2. Un caractère de l'intitulé sera considéré comme accélérateur clavier pour l'option de menu s'il est précédé du caractère tilde (~). Il apparaît en souligné lors de l'affichage du menu.
3. La modification d'un intitulé de menu par SETMDISTR n'est possible qu'avant l'affichage de la fenêtre contenant le menu. Elle doit donc être effectuée dans l'événement INIT de la fenêtre.

Exemple :

```
; Francisation de l'intitulé "Cascade"
SETMDISTR MDI_CASCADE%, " ~Chevauchement "
```

Voir aussi GETMDISTR\$, MDI_*%

Instruction MDIARRANGEWINDOWS (Librairie NSMisc)

Arrange les fenêtres filles d'une fenêtre MDI dans la surface de celle-ci.

Syntaxe	MDIARRANGEWINDOWS <i>handle-fenêtre, type</i>			
Paramètres	handle-fenêtre	POINTER	I	handle de la fenêtre mère MDI
	type	INT(4)	I	type d'arrangement demandé

1. Cette instruction ne s'applique qu'aux fenêtres de type MDIWINDOW.
2. type doit être l'une des constantes MDI_ARRANGE*% :
 - a) MDI_ARRANGE_CASCADE% : arranger en cascade
 - b) MDI_ARRANGE_HORZTILE% : arranger en mosaïque horizontale
 - c) MDI_ARRANGE_VERTTILE% : arranger en mosaïque verticale

Voir aussi MDI_ARRANGE*%

Constantes MDI_ARRANGE*% (Librairie NSMisc)

Constantes pour arranger des fenêtres filles MDI dans la surface de leur mère.

Syntaxe	Déclaration interne	Description
MDI_ARRANGE_CASCADE%		arranger en cascade
MDI_ARRANGE_HORZTILE%		arranger en mosaïque horizontale
MDI_ARRANGE_VERTTILE%		arranger en mosaïque verticale

Ces constantes sont utilisées en paramètre de l'instruction MDIARRANGEWINDOWS.

Voir aussi MDIARRANGEWINDOWS

Instruction MDISHOWHIDDENWINDOWS (Librairie NSMisc)

Permet de voir ou non la liste des fenêtres fille MDI cachées dans le menu fenêtres.

Syntaxe	MDISHOWHIDDENWINDOWS <i>showMode</i>			
Paramètre	showMode	INT(1)	I	mode d'affichage de la liste des fenêtres

Voir aussi MDIARRANGEWINDOWS , Constantes MDI_ARRANGE*%

Recherche d'un fichier dans une liste de répertoires

Fonction NSSEARCHPATHS\$ (Librairie NSMisc)

Recherche le fichier name\$ dans les répertoires indiqués par le paramètre paths\$. Le principal avantage de cette fonction est qu'elle permet d'éviter la taille limite de 255 caractères pour le chemin de recherche.

Syntaxe	NSSEARCHPATHS\$ (name\$, paths\$)		
Paramètres	name\$	CSTRING	nom du fichier recherché
	paths\$	CSTRING	noms des répertoires
Valeur retournée	CSTRING (255) retourne le chemin et le nom du fichier recherché. Cette chaîne est limitée à 255 caractères.		

1. Les noms des répertoires dans paths\$ sont séparés par des points-virgules. Les guillemets (") autour des noms de répertoires sont ignorés.
2. Le paramètre paths\$ peut être une CSTRING de plus de 255 caractères.
3. Il est possible de spécifier une ou plusieurs variables d'environnement contenant elles-mêmes un ou plusieurs noms de répertoires en mettant chaque nom de variable entre parenthèses (là aussi séparés par des points-virgules).
4. "(=EXE)" permet de spécifier le répertoire du programme en cours d'exécution.
5. Le paramètre PATH doit être constitué comme ceci :

```
h% =@STIN$
s$="(=&h%&)"
test$ = NSSEARCHPATHS$("MyFile.txt", s$)
```

6. La syntaxe (=entierPointeur) permet d'indiquer l'adresse d'une chaîne C de longueur quelconque contenant une liste de répertoires.

Exemple 1 :

```
Local test$
Local S$
Local STIN$(1000)
local pointer h%
INSERT AT END 'Search for nsdesign' TO LISTBOX1
test$ = NSSEARCHPATHS$("nsdesign.exe", "(path)")
INSERT AT END test$ TO LISTBOX1
INSERT AT END 'Search for Pour votre information.cov' TO LISTBOX1
test$ = NSSEARCHPATHS$("Pour votre information.cov", "C:\Documents and Settings\All
```

```
Users\Documents\Mes télécopies\Pages de garde communes")
INSERT AT END test$ TO LISTBOX1

INSERT AT END 'STIN$ < 255 Result <=255' TO LISTBOX1
STIN$ = "D:\tests\C'est un dossier pour tester les noms longs qui dépassent la fameuse
limite des 255 caractères de NS-DK utiliser la fonction nsSearchPaths et vérifier le
résultat\test\bmp\image\Nouveau dossier\Nouveau dossier2\Nouveau dossier3"
test$ = NSSEARCHPATHS$("toto.txt", STIN$)
INSERT AT END "Le fichier est sous" TO LISTBOX1
INSERT AT END test$ TO LISTBOX1
INSERT AT END "Length of test$ = "&& LENGTH test$ TO LISTBOX1

INSERT AT END 'STIN$ < 255 Result > 255 returns an empty string' TO LISTBOX1
test$ = NSSEARCHPATHS$("toto.txt", STIN$)
INSERT AT END "Le fichier est sous" TO LISTBOX1
INSERT AT END test$ TO LISTBOX1
INSERT AT END "Length of test$ = "&& LENGTH test$ TO LISTBOX1

INSERT AT END 'STIN$ > 255 Result <= 255' TO LISTBOX1
STIN$ = "(PATH);.;(=EXE);(NS-BIN);(NS-BCK);(NS-TXT);(NS-BMP);"
STIN$ = STIN$&"D:\tests\C'est un dossier pour tester les noms longs qui dépassent la
fameuse limite des 255 caractères de NS-DK utiliser la fonction nsSearchPaths et vérifier
le résultat\tesst\bmp\image\Nouveau dossier\Nouveau dossier2\Nouveau dossier3"

test$ = NSSEARCHPATHS$("toto.txt", STIN$)
INSERT AT END "Le fichier est sous" TO LISTBOX1
INSERT AT END test$ TO LISTBOX1
INSERT AT END "Length of test$ = "&& LENGTH test$ TO LISTBOX1

INSERT AT END 'STIN$ > 255 Result <= 255 using (=h%) works fine' TO LISTBOX1
STIN$ = "(PATH);.;(=EXE);(NS-BIN);(NS-BCK);(NS-TXT);(NS-BMP);"
STIN$ = STIN$&"D:\tests\C'est un dossier pour tester les noms longs qui dépassent la
fameuse limite des 255 caractères de NS-DK utiliser la fonction nsSearchPaths et vérifier
le résultat\test\bmp\image\Nouveau dossier\Nouveau dossier2\Nouveau dossier3"

h% =@STIN$
s$="(=&h%&)"
test$ = NSSEARCHPATHS$("toto.txt", s$)
INSERT AT END "Le fichier est sous" TO LISTBOX1
INSERT AT END test$ TO LISTBOX1
INSERT AT END "Length of test$ = "&& LENGTH test$ TO LISTBOX1
```

Exemple 2 :

```
Test$=nsSearchPaths$("monfichier.bmp", "(=EXE);(PATH);.;(NS-BIN)")
;Recherche le fichier monfichier.bmp dans les répertoires du programme en cours d'exécution
puis dans les répertoires du PATH, puis dans le répertoire courant (le point) et enfin
dans le (ou les) répertoire(s) indiqué(s) par la variable d'environnement NS-BIN.
```

Gestion des List Box

Il est possible avec la série de fonctions/instructions *EDITINPLACE d'afficher un contrôle dynamique provisoire à l'intérieur d'une List Box.

On peut par exemple afficher des Combo Box à l'intérieur des cellules de List Box. Les contrôles provisoires permettent d'enrichir l'IHM et de maximiser l'utilisation de l'espace client des fenêtres.

La première fonction à appeler est BEGINEDITINPLACE qui fait le travail de préparation, initialise la zone d'édition et retourne un handle dessus. Ce handle sera utilisé ensuite par les autres fonctions *EDITINPLACE.

Instruction **ADJUSTLISTBOXCOLUMNS** (Librairie NSMisc)

Cette instruction permet de définir la marge entre les colonnes de la List Box.

Syntaxe	ADJUSTLISTBOXCOLUMNS <i>lb, line%, nbl%, col%, nbc%, widthinc%</i>			
Paramètres	lb	CONTROL		nom de la List Box
	line%	INTEGER		première ligne à prendre en compte pour déterminer l'ajustement
	nbl%	INTEGER		nombre de lignes à ajuster
	col%	INTEGER		première colonne
	nbc%	INTEGER		nombre de colonnes à ajuster
	widthinc%	INTEGER		nombre de pixels entre les colonnes

Exemples :

```
ADJUSTLISTBOXCOLUMNS LB1, 0, LineCount%(LB1), 1, 3, 100
;Ajuste la largeur des colonnes 1 à 3 de la List Box LB1 en tenant compte du texte de ces colonnes pour toutes les lignes de la List Box et en ajoutant 100 ;pixels (marge entre les colonnes)
```

Site	Pays	Prix/Jour
Corfou	Grèce	500F
Vittel	France	850F
Djerba	Tunisie	800F
Djerba	Tunisie	800F
Djerba	Tunisie	800F
Djerba	Tunisie	800F
Djerba	Tunisie	800F

ADJUSTLISTBOXCOLUMNS LB1, 0, LineCount%(LB1), 1, 3, 10
;Ajuste la largeur des colonnes 1 à 3 de la List Box LB1 en tenant compte du texte de ces colonnes pour toutes les lignes de la List Box et en ajoutant 10 pixels (marge entre les colonnes)

Site	Pays	Prix/Jour
Corfou	Grèce	500F
Vittel	France	850F
Djerba	Tunisie	800F
Djerba	Tunisie	800F
Djerba	Tunisie	800F
Djerba	Tunisie	800F
Djerba	Tunisie	800F

Instruction GETLISTBOXCELL (Librairie NSMisc)

Calcule les coordonnées d'une cellule de la List Box ou d'un pixel selon le cas et stocke le résultat dans les champs X, Y, W, H, L et C du segment SEG_LBCELL.

Cette fonction est très utile pour faire de l'édition dans une zone de saisie dans une List Box.

Syntaxe	GETLISTBOXCELL lb, cell, xc%, yl%, flags%			
Paramètres	lb	CONTROL	I	nom de la List Box
	cell	SEGMENT	I/O	segment SEG_LBCELL
	xc%	INTEGER	I	coordonnée x d'un pixel ou numéro de la colonne
	yl%	INTEGER	I	coordonnée y d'un pixel ou numéro de la ligne
	flags%	INTEGER	I	constantes <u>LBCF</u> *%

1. Le segment SEG_LBCELL est défini dans le fichier NSMISC. NCL :

```
SEGMENT SEG_LBCELL ; List Box cell coordinates
INTEGER X ; pixel coords of the cell in the ListBox client area
INTEGER Y
INTEGER W
INTEGER H
INTEGER L ; Line of the cell
INTEGER C ; Column of the cell
```

```
INTEGER CLIP ; clipped sides of the cell (LBCC_%% combined)
ENDSEGMENT ; SEG_LBCELL
```

2. Les paramètres xc% et yl% indiquent :
 - a) soit les coordonnées (x, y) d'un pixel dans la zone cliente de la List Box, si le flag LBCF_FROMXY% est présent dans le paramètre flags%. Dans ce cas, on pourrait passer comme paramètres xc et xy les paramètres PARAM1% et PARAM2% des événements MOUSEMOVE, BUTTONDOWN, BUTTONUP et BUTTONDBLCLK de la List Box conviennent.
 - b) soit le numéro de la colonne (xc%) et de la ligne (yl%). Les colonnes et les lignes étant numérotées à partir de 0.
3. Les constantes LBCF_%% peuvent être ajoutées au paramètre flags%.
4. Le champ CLIP du segment SEG_LBCELL indique si un ou plusieurs côtés du rectangle sont coupés (au bord de la zone visible). Si le champ CLIP est différent de zéro, les bords du rectangle sont coupés.
5. Les constantes LBCC_%% correspondant au champ CLIP du paramètre cell.

Exemple :

```
LOCAL POINTER HDATA%

NEW SEG_LBCELL, HDATA%
IF HDATA%=0
  MESSAGE "ERREUR", "Handle de segment NULL"
  EXIT
ENDIF

FILL HDATA%, SIZEOF SEG_LBCELL, 0

GETLISTBOXCELL LB01, SEG_LBCELL(HDATA%), 1, 2, LBCF_CLIPCOORDS%

MOVE SEG_LBCELL(HDATA%).X TO EF_X
MOVE SEG_LBCELL(HDATA%).Y TO EF_Y
MOVE SEG_LBCELL(HDATA%).W TO EF_W
MOVE SEG_LBCELL(HDATA%).H TO EF_H
MOVE SEG_LBCELL(HDATA%).L TO EF_L
MOVE SEG_LBCELL(HDATA%).C TO EF_C
MOVE SEG_LBCELL(HDATA%).CLIP AND LBCC_CLIPRIGHT% TO EF_CLIP

DISPOSE HDATA%
```

Voir aussi BEGINEDITINPLACE, Constantes LBCC *, Constantes LBCF *

Instruction **ACTIVATEEDITINPLACE** (Librairie NSMisc)

Rend visible et active la zone d'édition du contrôle.

Syntaxe	ACTIVATEEDITINPLACE wnd, x%, y%		
Paramètres	wnd	POINTER	pointeur retourné par la

				fonction <u>BEGINEDITINPLACE</u>
	x%	INTEGER	I	coordonnée X
	y%	INTEGER	I	coordonnée Y

1. Les paramètres x% et y% doivent être supérieurs ou égaux à 0.
2. Le curseur de la zone de saisie est activé aux coordonnées X et Y.
3. Les coordonnées X, Y sont relatives au coin inférieur gauche du contrôle temporaire EditInPlace.
4. Si X ou Y valent 0 toute la zone du contrôle temporaire sera sélectionnée.
5. L'édition se termine lorsque :
 - a) le focus passe à une autre fenêtre (clic en dehors de l'éditeur, [Alt]+[Tab]...),
 - b) par pression de la touche [Echap] ou Entrée,
 - c) si le contrôle dans lequel la zone d'édition est attaché (ou sa fenêtre parente) reçoit un message VSCROLL, HSCROLL, CHANGED ou TERMINATE.
 - d) Dans tous les cas, sauf celui de la pression sur la touche [Echap], l'édition est considérée comme validée.

Exemple :

```

local POINTER HDATA%
local ret%
local retour%

NEW SEG_EIPINFO, HDATA%
IF HDATA%=0
  MESSAGE "ERREUR","Handle de segment NULL"
  EXIT
ENDIF

FILL HDATA%, SIZEOF SEG_EIPINFO, 0

MOVE 0 TO SEG_EIPINFO(HDATA%).VER
;The container control is the EntryField EF_TEST
@SEG_EIPINFO(HDATA%).CTRL=EF_TEST
MOVE 1 TO SEG_EIPINFO(HDATA%).X ; coordonnée gauche de la zone d'édition
MOVE 1 TO SEG_EIPINFO(HDATA%).Y ; coordonnée du bas de la zone d'édition
MOVE 68 TO SEG_EIPINFO(HDATA%).W ; largeur de la zone d'édition
MOVE 25 TO SEG_EIPINFO(HDATA%).H ; hauteur de la zone d'édition
;The temporary control is an EditComboBox DI_COMBBBOXE%
MOVE DI_COMBBBOXE% TO SEG_EIPINFO(HDATA%).kind
MOVE "Initial Value" to SEG_EIPINFO(HDATA%).text
ret%= BEGINEDITINPLACE (SEG_EIPINFO(HDATA%), 0)
move ret% to EF
IF ret%=0
  MESSAGE "ERREUR","Traitement terminé"
ELSE
  ; once the editin place has been called you can access the temporary Ctrl
  INSERT AT END 'Choix1' to SEG_EIPINFO(HDATA%).CTRL
  INSERT AT END 'Choix2' to SEG_EIPINFO(HDATA%).CTRL
  INSERT AT END 'Choix3' to SEG_EIPINFO(HDATA%).CTRL

```

```

;displays the temporary Control
ACTIVATEEDITINPLACE ret%,1,2
ENDIF
DISPOSE HDATA%

; The event User0 of the control EF_TEST
Insert at END "User0 of EF_TEST" to LISTBOX1
IF param12% <> -1
  IF param12% = 0
    Insert at END "Escape pressed" to LISTBOX1
  ELSE
    Insert at END "PARAM12$ "&&PARAM1$ to LISTBOX1
    Insert at END "PARAM12% "&&PARAM12% to LISTBOX1
  ENDIF
ENDIF
ENDIF

```

Voir aussi BEGINEDITINPLACE

Fonction BEGINEDITINPLACE (Librairie NSMisc)

Prépare l'édition d'une valeur dans une zone de saisie à l'intérieur d'un contrôle.

Syntaxe	BEGINEDITINPLACE (info, usermsg%)			
Paramètres	info	SEGMENT	I/O	segment SEG_EIPINFO
	usermsg%	INTEGER	I	numéro d'un message USERx
Valeur retournée	POINTER sur un contrôle EDITINPLACE temporaire			

1. Le segment SEG_EIPINFO est défini dans le fichier NSMISC.NCL :

```

SEGMENT SEG_EIPINFO ; Edit In Place Information
INTEGER VER ; Version = 0
CONTROL CTRL
INTEGER X ; abscisse du rectangle d'édition (relative au coin
; gauche du bas de CTRL)
INTEGER Y ; ordonnée du rectangle d'édition (relative au coin
; gauche du bas de CTRL)
INTEGER W ; Largeur du rectangle d'édition
INTEGER H ; Hauteur du rectangle d'édition
INTEGER KIND ; Kind of the editor control (DI_ENTRYFIELD%,
; DI_COMBBOX%, DI_COMBBOXE%)
CSTRING TEXT ; Init edit text (ENTRYFIELD, COMBBOXE)
ENDSEGMENT ; SEG_EIPINFO
;CTRL contient l'adresse du contrôle dans lequel on veut insérer un ;contrôle provisoire
KIND le type de contrôle provisoire qu'on ;souhaite afficher utiliser les constante DI_*
;TEXT le texte d'initialisation du contrôle provisoire

```

2. Les seuls types de contrôles temporaires admis dans la variable KIND du segment SEG_EIPINFO sont l'Entry-field, la Combo-box et la Combo-box

éditable (soient les constantes respectives suivantes : DI_ENTRYFIELD%, DI_COMBBOX% et DI_COMBBOXE%).

3. Le paramètre info doit être initialisé avec les valeurs requises avant d'appeler cette fonction.
4. Le paramètre usermsg% indique le numéro d'un message USERx pour notifier au contrôle conteneur que l'édition est terminée.
5. Si l'édition est terminée par une pression sur la touche [Echap], l'événement USERx est envoyé avec PARAM12% à 0, sinon soit PARAM1\$ contient le texte modifié, soit PARAM12% est le numéro de l'élément sélectionné si le contrôle provisoire est une Combo Box éditable.
6. Si le contrôle provisoire est un Entry Field ou une Combo Box éditable, il faut impérativement vérifier que PARAM12% est différent de -1 avant de lire PARAM1\$.
7. La fonction retourne un pointeur à utiliser comme paramètre wnd des autres fonction/instructions *EDITINPLACE sauf si sa valeur est 0 (erreur interdisant de poursuivre le traitement).
8. Au retour de l'appel à BEGINEDITINPLACE, la variable CTRL du segment SEG_EIPINFO est remplacé par le contrôle provisoire d'édition (du type indiqué dans Info.Kind), ce qui permet de l'initialiser (INSERT si c'est une Combo Box, éditable ou non, paramètres dynamiques CASE, CHARACTERS, FILLTEXT, FORMAT, JUSTIFICATION, MAXLEN, NOBLANKS, REQUIRED, SKIPBLANKS et/ou TYPE pour un Entry Field ou une Combo Box éditable).

Exemple :

```
local POINTER HDATA%
local ret%
local retour%

NEW SEG_EIPINFO, HDATA%
IF HDATA%=0
  MESSAGE "ERREUR","Handle de segment NULL"
  EXIT
ENDIF

FILL HDATA%, SIZEOF SEG_EIPINFO, 0

MOVE 0 TO SEG_EIPINFO(HDATA%).VER
;The container control is the EntryField EF_TEST
@SEG_EIPINFO(HDATA%).CTRL=EF_TEST
MOVE 1 TO SEG_EIPINFO(HDATA%).X ; coordonnée gauche de la zone d'édition
MOVE 1 TO SEG_EIPINFO(HDATA%).Y ; coordonnée du bas de la zone d'édition
MOVE 68 TO SEG_EIPINFO(HDATA%).W ; largeur de la zone d'édition
MOVE 25 TO SEG_EIPINFO(HDATA%).H ; hauteur de la zone d'édition
;The temporary control is an EditComboBox DI_COMBBOXE%
MOVE DI_COMBBOXE% TO SEG_EIPINFO(HDATA%).kind
MOVE "Initial Value" to SEG_EIPINFO(HDATA%).text

ret%= BEGINEDITINPLACE (SEG_EIPINFO(HDATA%), 0)
```

```

move ret% to EF
IF ret%=0
  MESSAGE "ERREUR","Traitement terminé"
ELSE
  ; once the editin place has been called you can access the temporary Ctrl
  INSERT AT END 'Choix1' to SEG_EIPINFO(HDATA%).CTRL
  INSERT AT END 'Choix2' to SEG_EIPINFO(HDATA%).CTRL
  INSERT AT END 'Choix3' to SEG_EIPINFO(HDATA%).CTRL
  ;displays the temporary Control
  ACTIVATEEDITINPLACE ret%,1,2
ENDIF
DISPOSE HDATA%

; The event User0 of the control EF_TEST
Insert at END "User0 of EF_TEST " to LISTBOX1
IF param12% <> -1
  IF param12% = 0
    Insert at END " Escape pressed" to LISTBOX1
  ELSE
    Insert at END "PARAM12$ "&&PARAM1$ to LISTBOX1
    Insert at END "PARAM12% "&&PARAM12% to LISTBOX1
  ENDIF
ENDIF
ENDIF

```

Voir aussi [GETLISTBOXCELL](#)

Fonction QUERYEDITINPLACE ([Librairie NSMisc](#))

Permet de récupérer le contenu du paramètre info à partir du pointeur wnd.

Syntaxe	QUERYEDITINPLACE (wnd, info)			
Paramètres	wnd	POINTER	I	pointeur retourné par la fonction <u>BEGINEDITINPLACE</u>
	info	SEGMENT	I/O	segment SEG_EIPINFO
Valeur retournée	INT(1)			

Le segment SEG_EIPINFO défini dans le fichier NSMISC.NCL :

```

SEGMENT SEG_EIPINFO ; Edit In Place Information
INTEGER VER ; Version = 0
CONTROL CTRL ; contient l'adresse du contrôle dans lequel on veut insérer un contrôle provisoire
INTEGER X ; abscisse du rectangle d'édition (relative au coin gauche du bas de CTRL)
INTEGER Y ; ordonnée du rectangle d'édition (relative au coin gauche du bas de CTRL)
INTEGER W ; Largeur du rectangle d'édition
INTEGER H ; Hauteur du rectangle d'édition
INTEGER KIND ; le type de contrôle provisoire qu'on souhaite afficher utiliser les constantes DI_*
CSTRING TEXT ; le texte d'initialisation du contrôle provisoire
ENDSEGMENT

```

Exemple :

```
Local SEG_EIPINFO INFO
LOCAL H%
local INT Retour%(1)
Insert at END "User0 du controle " to LISTBOX1
IF param12% <> -1
  IF param12% = 0
    Insert at END "On a appuyé sur la touche Escape " to LISTBOX1
  ELSE
    Insert at END "PARAM12$ "&&PARAM1$ to LISTBOX1
    Insert at END "PARAM12% "&&PARAM12% to LISTBOX1
  ENDIF
ENDIF
h% = @INFO
; getdata% returns the handle of the temporary control
; then QUERYEDITINPLACE fills the SEG_EIPINFO segment
retour% = QUERYEDITINPLACE (GETDATA%, SEG_EIPINFO(h%))
INSERT AT END 'X =' && INFO.X TO LISTBOX1
INSERT AT END 'Y =' && INFO.Y TO LISTBOX1
```

Voir aussi [BEGINEDITINPLACE](#)

Instruction **TERMINATEEDITINPLACE** ([Librairie NSMisc](#))

Referme la zone d'édition temporaire EDITINPLACE. Equivalent à l'appui sur la touche [Entrée] par l'utilisateur.

Syntaxe	TERMINATEEDITINPLACE <i>wnd, wantnewvalue%</i>			
Paramètres	wnd	POINTER	I	pointeur retourné par la fonction BEGINEDITINPLACE
	wantnewvalue%	INTEGER	I/O	indicateur booléen

Si wantnewvalue% est à TRUE%, l'événement USERx spécifié est envoyé, comme si l'édition avait été terminée normalement (validée).

Exemple :

```
local POINTER HDATA%
local ret%
local retour%

NEW SEG_EIPINFO, HDATA%
IF HDATA%=0
  MESSAGE "ERREUR","Handle de segment NULL"
  EXIT
ENDIF

FILL HDATA%, SIZEOF SEG_EIPINFO, 0

MOVE 0 TO SEG_EIPINFO(HDATA%).VER
;The container control is the EntryField EF_TEST
```

```

@SEG_EIPINFO(HDATA%).CTRL=EF_TEST
MOVE 1 TO SEG_EIPINFO(HDATA%).X ; coordonnée gauche de la zone d'édition
MOVE 150 TO SEG_EIPINFO(HDATA%).Y ; cordonnée du bas de la zone d'édition
MOVE 120 TO SEG_EIPINFO(HDATA%).W ; largeur de la zone d'édition
MOVE 40 TO SEG_EIPINFO(HDATA%).H ; hauteur de la zone d'édition
;The temporary control is an EditComboBox DI_COMBBBOXE%
MOVE DI_COMBBBOXE% TO SEG_EIPINFO(HDATA%).kind
MOVE "Initial Value" to SEG_EIPINFO(HDATA%).text

ret%= BEGINEDITINPLACE (SEG_EIPINFO(HDATA%), 0)
SETDATA ret%

move ret% to EF
IF ret%=0
  MESSAGE "ERREUR","Traitement terminé"
ELSE
  ; once the editin place has been called you can access the temporary Ctrl
  INSERT AT END 'Choix1' to SEG_EIPINFO(HDATA%).CTRL
  INSERT AT END 'Choix2' to SEG_EIPINFO(HDATA%).CTRL
  INSERT AT END 'Choix3' to SEG_EIPINFO(HDATA%).CTRL
  ;displays the temporary Control and select all the contents of the control
  ;if you want to place the cursor at a certain position in the temporary
  ;control specify the xy coordinates like ACTIVATEEDITINPLACE ret%,50,1
  ACTIVATEEDITINPLACE ret%,50,0
  ; Terminates the control after 1 second
  WAIT 1000
  TERMINATEEDITINPLACE ret%, TRUE%
ENDIF
DISPOSE HDATA%

```

Voir aussi [BEGINEDITINPLACE](#)

Constantes LBCC_*(Librairie NSMisc)

Valeurs représentant le format du rectangle de la zone de saisie intégrée à une List Box.

Syntaxe	Déclaration interne	Description
LBCC_CLIPLEFT%	\$01	Bord gauche coupé.
LBCC_CLIPRIGHT%	\$02	Bord droit coupé.
LBCC_CLIPTOP%	\$04	Haut coupé.
LBCC_CLIPBOTTOM%	\$08	Bas coupé.
LBCC_CLIPALL%	\$10	Coupé dans tous les côtés.
LBCC_BELOWLASTLINE%	\$20	Coupé après la dernière ligne.
LBCC_RIGHTOFLASTCOL%	\$40	Coupé à droite du bord droit.

LBCC_TOPLOCKED%	\$80	Valeur retournée uniquement. La cellule fait partie des lignes d'entêtes (bloquées) d'une List Box.
-----------------	------	---

Ces constantes sont utilisées par le champ CLIP du segment SEG_LBCELL dans l'instruction GETLISTBOXCELL.

Voir aussi GETLISTBOXCELL

Constantes LBCF_**% (Librairie NSMisc)

Valeurs représentant le format de la zone de saisie intégrée à une List Box.

Syntaxe	Déclaration interne	Description
LBCF_FROMXY%	\$01	Les paramètres xc% et yl% de la fonction GETLISTBOXCELL correspondent aux coordonnées x/y d'un pixel.
LBCF_CLIPCOORDS%	\$02	coupe le rectangle restitué aux bords de la zone visible de la List Box.
LBCF_ENLARGERIGHTCOL%	\$04	si la cellule indiquée est dans la dernière colonne, agrandit (si possible) celle-ci jusqu'au bord droit de la zone visible.
LBCF_WHOLELINE%	\$08	rend un rectangle englobant toute la ligne (ou toute la largeur de la zone visible si LBCF_CLIPCOORDS% est présent).

Ces constantes sont utilisées par le dernier paramètre flags% de l'instruction GETLISTBOXCELL.

Voir aussi GETLISTBOXCELL

Manipulation des listes d'images et affectation d'images aux items de menu

Nat System a décidé de publier l'interface de programmation Window Image List (de la bibliothèque de service NSMISC) pour vous permettre d'associer des images ou bitmaps aux choix de menus.

Cette nouvelle interface s'appuie sur la création, la mise à jour, et la manipulation de liste d'images attachées aux fenêtres appelées Window Image Lists ou WIL. Les formats d'images acceptés sont le JPEG, GIF et BMP (Les JPEG ne permettent pas une couleur de transparence).

Les listes d'images seront désignées dans la suite de ce document par le terme WIL. ||

Les WIL sont des listes de bitmap de taille identique. On ne peut pas avoir des images de tailles différentes dans une même WIL.

Il sera possible de créer plusieurs WIL et de les attacher à une fenêtre mais c'est uniquement à partir de la première WIL d'index 0 qu'on pourra affecter des images à des items de menu. Les WIL d'index supérieur à 0 pourront servir pour des dessins dans un PS% (Presentation Space) mais pas pour enrichir en images les items de menu.

Fonction GETMENUITEMPICTURE% (Librairie NSMisc)

Retourne l'index de l'image associée à un item de menu ou un chiffre compris entre \$F000 et \$F014 si l'image est une des images prédéfinies du runtime ou \$FFFF dans le cas d'absence d'image associée aux items du menu.

Syntaxe	GETMENUITEMPICTURE% (<i>wnd</i> , <i>Id%</i>)			
Paramètres	wnd	POINTER	I	handle de la fenêtre
	Id%	INTEGER	I	identifiant de l'item de menu
Valeur retournée	INT(2)			

1. La plage \$F000...\$FFFE est réservée pour des images du runtime. Actuellement une petite partie de cette plage comprise entre \$F000 et \$F014 comporte des images comme indiqué dans l'introduction. En dessous de cette plage ce sont les index des images de la première WIL de la fenêtre (ou de sa fenêtre mère si c'est une fille MDI et qu'elle n'a pas sa propre WIL) qui seront retournées par GETMENUITEMPICTURE%.
2. L'identifiant du menu item peut être obtenu en appelant la fonction GETCONTROLID%.

3. Cette fonction est équivalente à l'appel de la propriété dynamique Picture du menuItem :

```
i% = MenuItem.picture
```

Exemple :

```
local i%(2)
i%=GetMenuItemPicture (self%,GETCONTROLID%(MNU_DEL))
SetMenuItemPicture (self%,GETCONTROLID%(MNU_CLOSE), i%)
```

Voir aussi GETCONTROLID%, WIL_APPENDFROMBMP%

Instruction SETMENUITEMPICTURE (Librairie NSMisc)

Fixe l'image d'un item de menu.

Syntaxe	SETMENUITEMPICTURE <i>wnd, Id%, pict%</i>		
Paramètres	wnd	POINTER	I handle de la fenêtre
	Id%	INTEGER	I identifiant de l'item de menu
	pict%	INT(2)	I index ou identifiant de l'image

1. L'identifiant du menu item peut être obtenu en appelant la fonction GETCONTROLID%.
2. Cette fonction est équivalente à l'appel de la propriété dynamique Picture du menuItem :

```
MnuItem.picture = $F014
```

Exemple :

```
;First image of the first WIL
SetMenuItemPicture (self%,GETCONTROLID%(MNU_CLOSE), 0)
;Runtime Image $F00C
SetMenuItemPicture (self%,GETCONTROLID%(MNU_ABOUT), $F00C)
;Second image of the first WIL
SetMenuItemPicture (self%,GETCONTROLID%(MNU_OUVR), 1)
```

Voir aussi GETCONTROLID%, WIL_APPENDFROMBMP%

Fonction WIL_APPENDFROMBMP% (Librairie NSMisc)

Ajoute une ou plusieurs images à la fin d'une Image List. Si l'Image List n'existe pas encore, elle est créée.

Syntaxe	WIL_APPENDFROMBMP% (<i>wnd</i> , <i>Index%</i> , <i>Bmp%</i> , <i>PicturesWidth%</i> , <i>TranspColor%</i>)			
Paramètres	wnd	POINTER	I	handle de la fenêtre
	index%	INTEGER	I	index de la WIL. (ou -1 si on veut ajouter une nouvelle WIL après celles qui existent déjà pour la fenêtre de handle wnd)
	bmp%	POINTER	I	handle de la bitmap contenant la ou les images. (la hauteur est celle de la WIL, la largeur doit être un multiple de la largeur des images de la WIL)
	PicturesWidth%	INTEGER	I	la largeur des images en pixels (doit être identique à celles des images de la WIL si elles existent déjà)
	TranspColor%	INTEGER	I	la couleur de transparence.
Valeur retournée	La valeur retournée est soit l'index de la première image ajoutée à la WIL (0 si elle est initialement vide ou inexistante), soit l'index de la nouvelle WIL si le paramètre donnant l'index de la WIL était égale à -1 (ajout d'une nouvelle WIL, dans ce cas, sa première image est toujours à l'index 0). Retourne -1 en cas d'erreur.			

1. La couleur transparente est un entier 32 bits, si l'octet de poids fort est à 0, c'est une des couleurs COL_*, les valeurs -4...-1 indiquent qu'il faut utiliser la couleur du pixel inférieur droit/supérieur droit/inférieur gauche/supérieur gauche de la première image de la bitmap, -5 indique que les images sont opaques et une valeur du type \$FErrvvbb permet de spécifier une couleur RVB où rr, vv et bb sont les digits hexadécimaux correspondants aux composantes rouge, verte et bleu.

2. Il est possible depuis la x.61 de traiter les formats d'image JPEG et GIF en plus des BMP, toutefois il n'est pas possible d'avoir une couleur de transparence avec les JPEG.

Exemple :

```
Local hbmp%, index%
;this bmp has height of 32 and a width of 3x32 pixels
;therefore it makes a WIL of 3 bitmaps
MOVE CREATEBMP% ("(NS-BMP)\Natsystem.BMP") TO HBMP% ;Creates the bitmap hBMP%
index% = WIL_AppendFromBmp%(self%,-1,HBMP%,32,-1)
index% = WIL_GetCount%(self%)
insert AT END "le nombre de liste ="&&index% to MLE
;a new WIL with Gifs
MOVE CREATEBMP% ("(NS-BMP)\Natsystem.gif") TO HBMP%
index% = WIL_AppendFromBmp%(self%,-1,HBMP%,32,-1)
index% = WIL_GetCount%(self%)
insert AT END "le nombre de liste ="&&index% to MLE
```

Voir aussi [WIL_FINDORADDONEPICTURE%](#), [WIL_APPENDFROMMEM%](#)

Instruction **WIL_DELETEPICTURES** (Librairie NSMisc)

Retire des images d'une WIL. Les images qui suivent les images supprimées ont leur index diminué d'autant.

Syntaxe	WIL_DELETEPICTURES <i>wnd, Index%, First%, Count%</i>		
Paramètres	wnd	POINTER	handle de la fenêtre
	Index%	INTEGER	index de la WIL.
	First%	INTEGER	index de la première image
	Count%	INTEGER	nombre d'images à supprimer

Exemple :

```
; deletes the first image from the first WIL
INTEGER COUNT%
WIL_DeletePictures self%,0,0,1
```

Voir aussi [WIL_APPENDFROMBMP%](#), [WIL_FINDORADDONEPICTURE%](#)

Instruction **WIL_DESTROY** (Librairie NSMisc)

Détruit la WIL indiquée. Les WIL suivantes ne sont pas décalées (leur index reste le même, l'index libéré peut être réutilisé pour une nouvelle WIL).

Syntaxe	WIL_DESTROY <i>wnd, Index%</i>
---------	---------------------------------------

Paramètres	wnd	POINTER	I	handle de la fenêtre
	Index%	INTEGER	I	index de la WIL

Exemple :

```
local nb%
nb% = WIL_GetCount%(self%)
if nb% > 0
WIL_DESTROY(self%,nb% - 1) ; destroy the last WIL
endif
```

Voir aussi WIL_APPENDFROMBMP%, WIL_GETCOUNT%

Instruction **WIL_DRAWPICTURE** (Librairie NSMisc)

Dessine l'image demandée.

Syntaxe	WIL_DRAWPICTURE <i>ps%, x%, y%, Style%, wnd, IndexIL%, IndexPict%</i>			
Paramètres	ps%			Presentation Space
	x%	INTEGER	I	coordonnées x du coin inférieur gauche de l'image
	y%	INTEGER	I	coordonnées y du coin inférieur gauche de l'image
	style%	INTEGER	I	mode de dessin constante ILD_TRANSPARENT% avec le bit de transparence pris en compte ou ILD_MASK% pour dessiner la silhouette de l'image.
	wnd	POINTER	I	handle de la fenêtre
	IndexIL%	INTEGER	I	index de la WIL
	IndexPict%	INTEGER	I	index de l'image

Exemple :

```
INIT Main Window
GLOBAL POINTER handleFille%
GLOBAL int gPaint%(1)
gPaint% = False%
;Better to leave the window get initialized first
post WIL
```

```

user Event WIL of the main WINDOW
Local hbmp%, index%
MOVE CREATEBMP% ("(NS-BMP)\MTINTIN.BMP") TO HBMP% ;Creates the bitmap BMP%
MOVE HBMP% TO BMP_SA ; Displays the bitmap
index% = WIL_AppendFromBmp%(self%,-1, HBMP%,32,-1) ; a list of 4 pictures
OpenH WDESSIN, 0, handleFille%
ATTACHCHILDWINDOW SELF%, CHILD_TOPAREA% , handleFille%
gPaint% = TRUE%

;Event Paint of childWindow WDESSIN
; it draws all the pictures of all the WILs attached to the parentWindow
LOCAL POINTER PS%, i%, j%
LOCAL INTEGER X%
LOCAL INTEGER Y%
local INTEGER width%, INTEGER height%
MOVE PARAM12% TO PS%
GPI_ERASE PS%
if gpaint%
  X% = 0; getclientwidth%
  for i% = 0 to wil_getcount% (parentwindow%) -1
    WIL_GetSize parentwindow%, i%, width%, height%
    Y% = getclientheight%/2 - height%/2
    for j% = 0 to wil_getpicturescount% (parentwindow%, i%) -1
      ; the Y is calculated to be the center
    ; while the X is incremented
      wil_DrawPicture PS%, X%,Y%,0,parentwindow%, i%, j%
    X% = X% + width%
  endfor
endfor
endif

```

Fonction WIL_FINDORADDONEPICTURE% (Librairie NSMisc)

Ajoute une ou plusieurs images à la fin d'une WIL.

Syntaxe	WIL_FINDORADDONEPICTURE% (wnd, Index%, Bmp%, TranspColor%, ppHash)		
Paramètres	wnd	POINTER	handle de la fenêtre
	index%	INTEGER	index de la 'WIL. (ou -1 si on veut ajouter une nouvelle WIL après celles qui existent déjà pour cette fenêtre)
	bmp%	POINTER	handle de la bitmap

			contenant la ou les images. (la hauteur est celle de la WIL, la largeur doit être un multiple de la largeur des images)
	PicturesWidth%	INTEGER	I la largeur des images (en pixels, doit être identique à celles de la WIL si elle existe déjà)
	TranspColor%		I la couleur devant être considérée comme transparente.
Valeur retournée	Retourne l'index de l'image trouvée (si elle est déjà présente) ou ajoutée, ou -1 en cas d'erreur.		

La couleur transparente est un entier 32 bits, si l'octet de poids fort est à 0, c'est une des couleurs COL_*, les valeurs -4...-1 indiquent qu'il faut utiliser la couleur du pixel inférieur droit/supérieur droit/inférieur gauche/supérieur gauche de la première image de la bitmap, -5 indique que les images sont opaques et une valeur du type \$FErrvbb permet de spécifier une couleur RVB où rr, vv et bb sont les digits hexadécimaux correspondants aux composantes rouge, verte et bleu.

Exemple :

```
Local hbmp%, index%
Local POINTER pthash
pthash = 0
MOVE CREATEBMP% ("NS-BMP)\validoff.BMP") TO HBMP%
; finds the first occurrence of the picture
index% = WIL_FindOrAddOnePicture% (self%,0,HBMP%,-1, @pthash)
if index% = -1
```

```
insert AT END "une erreur c'est produite" to MLE
else
insert AT END "l'image est à la "&&index% &&"place" to MLE
endif
WIL_FreeHash pthash
```

Voir aussi WIL_FREEHASH, WIL_FINDPICTURE%

Fonction WIL_FINDPICTURE% (Librairie NSMisc)

Recherche une image dans une WIL (pixels identiques) et retourne l'index de la première image qui correspond aux critères de la recherche ou -1 en cas d'échec.

Syntaxe	<u>WIL_FINDPICTURE%</u> (<i>Wnd, Index%, Bmp%, TranspColor%, ppHash</i>)			
Paramètres	wnd	POINTER	I	handle de la fenêtre
	index%	INTEGER	I	index de la WIL
	bmp%	INTEGER	I	handle de la bitmap à rechercher (elle doit avoir les dimensions des images de la WIL)
	TranspColor%	INTEGER	I	la couleur devant être considérée comme transparente.
	ppHash	POINTER	I	pointeur (passé par adresse, modifié par la fonction) qui doit être mise à 0 (NULL) avant une première recherche dans une WIL et passé à <u>WIL_FREEHASH</u> après la dernière recherche ou si la WIL est modifiée autrement que par <u>WIL_FINDORADDDONEPICTURE%</u> (utilisé pour optimiser la vitesse des comparaisons lorsque plusieurs images sont recherchées/ajoutées).
Valeur retournée	retourne l'index de la première image qui correspond aux critères de la recherche ou -1 en cas d'échec.			

La couleur transparente est un entier 32 bits, si l'octet de poids fort est à 0, c'est une des couleurs COL_*, les valeurs -4...-1 indiquent qu'il faut utiliser la couleur du pixel inférieur droit/supérieur droit/inférieur gauche/supérieur gauche de la première image de la bitmap, -5 indique que les images sont opaques et une valeur du type \$FErrvbb permet de spécifier une couleur RVB où rr, vv et bb sont les digits hexadécimaux correspondants aux composantes rouge, verte et bleu.

Exemple :

```
Local hbmp%, index%
Local POINTER pthash
pthash = 0
MOVE CREATEBMP% ("NS-BMP)\validoff.BMP") TO HBMP%
index% = WIL_FindPicture% (self%,0,HBMP%,-1, @pthash)
if index% = -1
    insert AT END "l'image n'a pas été trouvée" to MLE
else
    insert AT END "l'image est à la "&&index% &&"place" to MLE
endif
WIL_FreeHash pthash
```

Voir aussi WIL_FREEHASH

Instruction WIL_FREEHASH (Librairie NSMisc)

Libère la mémoire allouée par WIL_FINDPICTURE% ou par WIL_FINDORADDONEPICTURE%.

Syntaxe	WIL_FREEHASH pHash			
Paramètre	pHash	POINTER	I/O	pointeur passé en dernier paramètre d'une de ces deux fonctions <u>WIL_FINDPICTURE%</u> ou par <u>WIL_FINDORADDONEPICTURE%</u> .

Exemple :

```
Local hbmp%, index%
Local POINTER pthash
pthash = 0
MOVE CREATEBMP% ("NS-BMP)\validoff.BMP") TO HBMP%
index% = WIL_FindPicture% (self%,0,HBMP%,-1, @pthash)
if index% = -1
    insert AT END "l'image n'a pas été trouvée" to MLE
else
    insert AT END "l'image est à la "&&index% &&"place" to MLE
endif
WIL_FreeHash pthash
```

Voir aussi WIL_FINDPICTURE%, WIL_FINDORADDONEPICTURE%

Fonction **WIL_GETCOUNT%** (Librairie NSMisc)

Retourne le nombre de WIL associées à une fenêtre. Une WIL contient une ou plusieurs images de même taille (16x16 pixels en général, mais pas nécessairement) avec masque de transparence (comme les icônes).

Syntaxe	WIL_GETCOUNT% (<i>wnd</i>)		
Paramètre	wnd	POINTER	I handle de la fenêtre
Valeur retournée	INTEGER Nombre de WIL pour la fenêtre.		

On peut associer une ou plusieurs WIL à une fenêtre.

Exemple :

```
local index%
index% = WIL_GetCount%(self%)
insert AT END "le nombre de liste ="&&index% to MLE
```

Voir aussi [WIL_APPENDFROMBMP%](#)

Fonction **WIL_GETPICTURESCOUNT%** (Librairie NSMisc)

Retourne le nombre d'images d'une WIL d'une fenêtre.

Syntaxe	WIL_GETPICTURESCOUNT% (<i>wnd</i> , <i>Index%</i>)		
Paramètres	wnd	POINTER	I handle de la fenêtre
	index%	INTEGER	I index de la WIL (de 0 à WIL_GetCount% (Self%) - 1).
Valeur retournée	INTEGER Le nombre d'images de la WIL.		

Exemple :

```
INSERT AT END "Le nombre d'images dans la 1 ère liste est " & \
WIL_GetPicturesCount%(self%, 0) TO MLE
```

Voir aussi [WIL_APPENDFROMBMP%](#), [WIL_GETCOUNT%](#)

Instruction **WIL_GETSIZE** (Librairie NSMisc)

Rend les dimensions des images de la WIL indiquée.

Syntaxe	WIL_GETSIZE <i>wnd, Index%, Width%, Height%</i>			
Paramètres	wnd	POINTER	I	handle de la fenêtre
	Index%	INTEGER	I	index de la WIL.
	Width%	INTEGER	I/O	largeur des images de la WIL.
	Height%	INTEGER	I/O	hauteur des images de la WIL

Exemple :

```
local INTEGER index%, INTEGER width%, INTEGER height%
index% = 0 ; first WIL
WIL_GetSize self%, index%, width%, height%
insert AT END "la largeur des images de la WIL "&index%&&"est de "&width%&&" leur hauteur
de "&height% to MLE
```

Instruction WIL_SETCOUNT (Librairie NSMisc)

Fixe le nombre de WIL d'une fenêtre. L'emploi est optionnel, les fonctions qui ajoutent des images dans une WIL la crée si besoin est.

Syntaxe	WIL_SETCOUNT <i>wnd, Count%</i>			
Paramètres	wnd	POINTER	I	handle de la fenêtre
	count%	INTEGER	I	nombre d'Image List

Si le nombre d'Image List est inférieur à WIL_GETCOUNT%(Self%), les Images List en trop seront détruites.

Exemple :

```
local index%
WIL_SetCount(self%, 1)
index% = WIL_GetCount%(self%)
insert AT END "le nombre de liste ="&&index% to MLE ;1
```

Voir aussi WIL_APPENDFROMBMP%, WIL_GETCOUNT%

Fonction **WIL_APPENDFROMMEM%** (Librairie NSMisc)

Ajoute une ou plusieurs images à la fin d'une Image List. Si l'Image List n'existe pas encore, elle est créée.

Syntaxe	WIL_APPENDFROMMEM% <i>wnd, Index%, pMem, Size%, PicturesWidth%, TranspColor%</i>		
Paramètres	wnd	POINTER	handle de la fenêtre
	Index%	INTEGER	index de la WIL.
	pMem	POINTER	pointeur sur un bloc mémoire
	Size%	INTEGER	nombre d'octets du bloc mémoire
	PicturesWidth%	INTEGER	la largeur des images en pixels (doit être identique à celles des images de la WIL si elles existent déjà)
	TranspColor%	INTEGER	la couleur de transparence.
Valeur retournée	INTEGER		

1. Utilise le décodeur d'images des bibliothèques portables pour transformer les octets en image (BMP au format Windows reconnues, ainsi que les images GIF ou JPEG en Win32).
2. La couleur transparente est un entier 32 bits, si l'octet de poids fort est à 0, c'est une des couleurs COL_*, les valeurs -4...-1 indiquent qu'il faut utiliser la couleur du pixel inférieur droit/supérieur droit/inférieur gauche/supérieur gauche de la première image de la bitmap, -5 indique que les images sont opaques et une valeur du type \$FErrvvbb permet de spécifier une couleur RVB où rr, vv et bb sont les digits hexadécimaux correspondants aux composantes rouge, verte et bleu.

Voir aussi WIL_APPENDFROMBMP%, WIL_FINDORADDONEPICTURE%

Constantes SMIP_*% (Librairie NSMisc)

Ces constantes permettent d'affecter dynamiquement des images prédéfinies dans le runtime Nat System aux items de menu.

Syntaxe	Déclaration interne	Description
SMIP_FirstStdPict% ou SMIP_FileNew%	\$F000	nouveau
SMIP_FileNew2%	\$F001	nouveau document (variante de \$F000)
SMIP_FileOpen%	\$F002	ouverture
SMIP_FileSave%	\$F003	sauvegarde
SMIP_FileSaveAll%	\$F004	sauvegarde tous les documents
SMIP_FilePrint%	\$F005	impression
SMIP_EditUndo%	\$F006	annuler une action
SMIP_EditRedo%	\$F007	refaire une action
SMIP_EditCut%	\$F008	couper
SMIP_EditCopy%	\$F009	copier
SMIP_EditPaste%	\$F00A	coller
SMIP_EditDelete%	\$F00B	supprimer
SMIP_Help%	\$F00C	aide
SMIP_Search%	\$F00D	recherche

SMIP_FindNext%	\$F00E	recherche de l'élément suivant
SMIP_FindPrev%	\$F00F	recherche de l'élément précédent
SMIP_WinCascade%	\$F010	fenêtres en cascade (MDI)
SMIP_WinSplitHorz%	\$F011	fenêtres en mosaïque horizontale (MDI)
SMIP_WinSplitVert%	\$F012	fenêtres en mosaïque verticale (MDI)
SMIP_WinChoose%	\$F013	sélection d'une fenêtre (MDI)
SMIP_ExitApp%	\$F014	fermeture de l'application
SMIP_FileClose1%	\$F015	fermeture d'un fichier
SMIP_FileClose2%	\$F016	fermeture d'un fichier
SMIP_FileDelete%	\$F017	suppression d'un fichier
SMIP_Zoom%	\$F018	zoom
SMIP_Tools%	\$F019	outils
SMIP_HelpContent%	\$F01A	contenu de l'aide
SMIP_HelpIndex%	\$F01B	index de l'aide
SMIP_HelpSearch%	\$F01C	recherche dans l'aide

1. Pour pouvoir associer des images aux items de menu il ne faut pas que la variable NoMenuPictures soit égale à True dans le fichier NSLIB.INI.
2. Ces images ont une taille unique de 16x16 pixels, c'est la taille conseillée pour des images de menu.

Exemple :

```
MnuItem.picture = SMIP_FileNew%
;Où MnuItem est le nom d'un item de menu et SMIP_FileNew% la référence de l'image
« nouveau document » du runtime.
```

Voir aussi [WIL_FINDORADDONEPICTURE%](#), [WIL_APPENDFROMBMP%](#), *paramètre dynamique* [.PICTURE](#)

Gestion de répertoires temporaires

Gestion de répertoires temporaires ([Librairie nsMisc](#))

Il est possible d'associer des répertoires temporaires spécifiques à un exécutable.

fonction GetTmpDir\$

instruction CleanTmpDir

Fonction getTmpDir\$ (Librairie nsMisc)

Retourne le nom d'un répertoire temporaire spécifique à l'exécutable courant

Syntaxe	Function getTmpDir\$ (rootDir\$, usage\$)			
Paramètres	cString	rootDir\$		répertoire racine.
	cString	usage\$		nom du sous-répertoire temporaire spécifique.
Valeur retournée	cString	nom complet du répertoire temporaire spécifique.		

Cette fonction effectue plusieurs actions.

1. Création du sous-répertoire "tmpNS" comme sous-répertoire du répertoire racine.
2. Création du sous-répertoire usage\$ comme sous répertoire du précédent.
3. La fonction renvoie la concaténation des deux répertoires précédents.

Exemple :

Cet exemple crée un répertoire temporaire composé de "myprog_" et de l'heure courante exprimée en secondes. Puis y crée un fichier "myFic.txt"

```
local tmpD$  
  
tmpD$ = GetTmpDir$("(TMP)", "myProg_" & CURRENTTIME%)  
h% = t_CREATE%(tmpD$ & "\myFic.txt")  
T_WriteLn (h%,"First line")  
T_Close (h%)
```

Note

Quand le programme se termine, le, ou les répertoires temporaires créés par getTmpDir\$ sont automatiquement effacés, s'il n'y en a pas eu d'autre de créés sous le répertoire tmpNS, celui-ci est également effacé.

Voir aussi [CleanTmpDir](#)

Instruction cleanTmpDir (Librairie nsMisc)

Efface le contenu d'un répertoire temporaire spécifique, s'il n'y en a pas eu d'autre de créés sous le répertoire tmpNS, celui-ci est également effacé.

Syntaxe	Instruction cleanTmpDir (usage\$)		
Paramètres	cString	usage\$	I nom du sous-répertoire temporaire spécifique.

Note

Quand le programme se termine, le répertoire temporaire est également automatiquement effacé.

Voir aussi [getTmpDir\\$](#)

Gestion des traces

Instruction SetNSGenTrace (Librairie NSMisc)

Permet de modifier le formatage des traces en fournissant un pointeur sur une fonction MyTrace.

Syntaxe	SETNSGENTRACE <i>traceFunc</i>		
Paramètre	traceFunc	POINTER	I pointeur vers une fonction MyTrace

Voir aussi [MyTrace](#)

Instruction MyTrace (Librairie NSMisc)

Permet de modifier le formatage des traces.

Syntaxe	MYTRACE <i>trace, line%, info</i>		
Paramètres	trace	SEGMENT	I référence sur le segment SEG_NSAGENTRACE

	line%	INT		numéro de ligne du fichier source
	info	POINTER		non utilisé actuellement

1. Le paramètre line% correspond au numéro de ligne où s'est produit l'événement de trace. La première ligne de l'événement d'une fenêtre est 1.
2. Le pointeur info est réservé pour un usage ultérieur et pourrait pointer sur des informations supplémentaires. Actuellement, il est nul pour toutes les traces.

Voir aussi SetNSGenTrace

Instruction NS_TRACE (Librairie NSMisc)

Permet d'intégrer des traces utilisateur dans les programmes NCL.

Syntaxe	NS_TRACE message\$		
Paramètre	message\$	CSTRING	traces utilisateur

1. Par défaut le formatage des traces suit l'exemple suivant (en-tête compris) :

```

NSDK:14:32:49 *****
NSDK:14:32:49 NSGENTRACE-<TRC> { ID in BINARY/FILE(LINE)
NSDK:14:32:49 NSGENTRACE-<TRC> } ID in BINARY/FILE(LINE) (CLOCK=CPU)
NSDK:14:32:49                                     \_____ CPU time between
NSDK:14:32:49                                     \_____ enter and leave
NSDK:14:32:49                                     \_____ line in FILE
NSDK:14:32:49                                     \_____ ressource FILE
NSDK:14:32:49                                     \_____ BINARY (dll or exe)
NSDK:14:32:49                                     \_____ ID of function/instruction/event
NSDK:14:32:49                                     \_____ Enter or leave funct/instr/event
NSDK:14:32:49 \_____ Generated trace reason 'instr', 'funct', 'event' for
NSDK:14:32:49 \_____ respectively instruction, function and control event
NSDK:14:32:49 *****
NSDK:14:32:49 NSGENTRACE-event { SEQ.EXECUTED in TEST.EXE/TEST.SCR(1)
NSDK:14:32:49 NSGENTRACE-instr  { G03 in TEST.EXE/SEQ.NCL(137)
NSDK:14:32:50 NSGENTRACE-instr  } G03 in TEST.EXE/SEQ.NCL(139) (CLOCK=0001.047)
NSDK:14:32:50 NSGENTRACE-event } SEQ.EXECUTED in TEST.EXE/TEST.SCR(2) (CLOCK=0001.047)
NSDK:14:32:55 NSGENTRACE-event { <CLIENT>.TERMINATE in TEST.EXE/TEST.SCR(1)
NSDK:14:32:55 NSGENTRACE-event } <CLIENT>.TERMINATE in TEST.EXE/TEST.SCR(2)
(CLOCK=0000.000)

```

2. La syntaxe NS_TRACE var\$ autorise désormais une chaîne de caractères supérieure à 255 caractères. Cependant, la syntaxe NS_TRACE var1\$&var2\$ est toujours limitée à 255 caractères.

Voir aussi Segment SEG NSGENTRACE, Constantes GTE **%

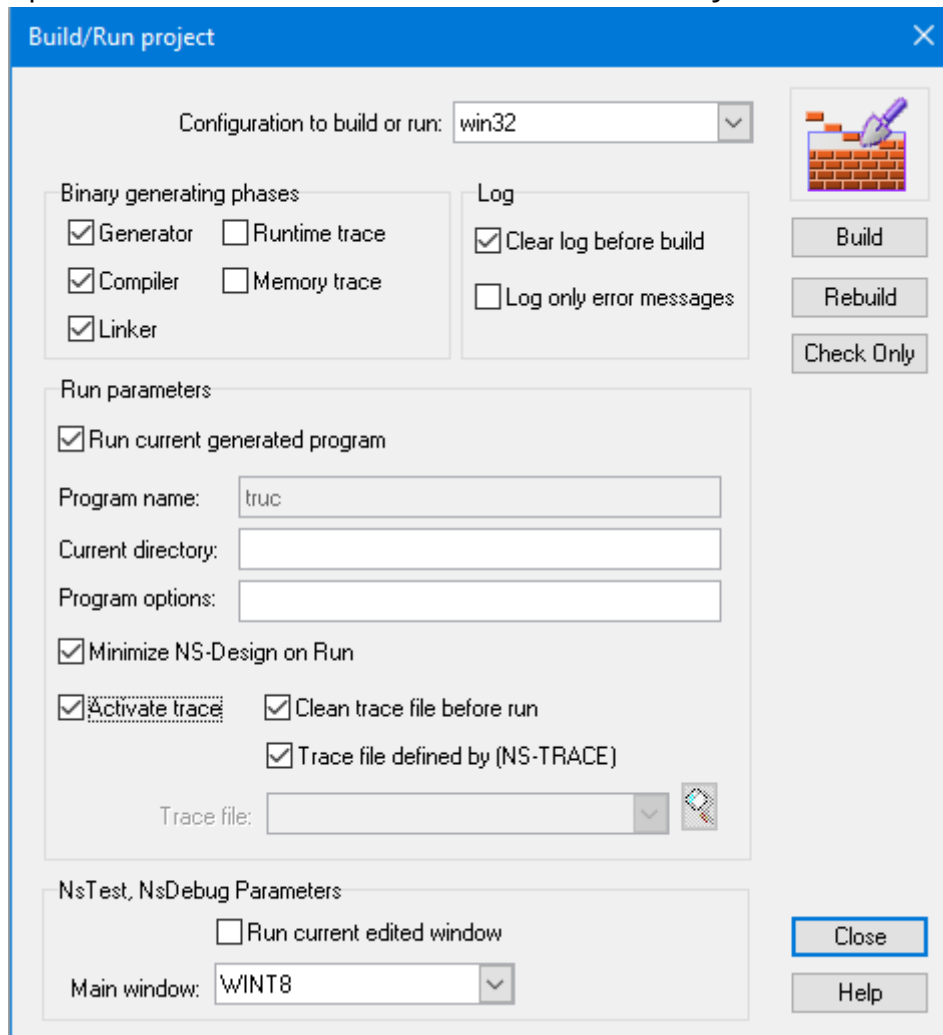
Pour activer la trace, il faut définir la variable d'environnement NS-TRACE à un nom de fichier.

```
set NS-TRACE=d:\tmp\myTrace.txt
```

cette version fonctionne pour les exécutables.

trace en mode NS-TEST

pour activer les traces en mode interprété de NS-DK (NS-TEST), il faut activer les options "Activate Trace" et "Trace File defined by (NS-TRACE)"



Build/Run project

Configuration to build or run: win32

Binary generating phases

- ☒ Generator ☐ Runtime trace
- ☒ Compiler ☐ Memory trace
- ☒ Linker

Log

- ☒ Clear log before build
- ☐ Log only error messages

Run parameters

- ☒ Run current generated program
- Program name: truc
- Current directory:
- Program options:
- ☒ Minimize NS-Design on Run
- ☒ Activate trace ☒ Clean trace file before run
- ☒ Trace file defined by (NS-TRACE)
- Trace file:

NsTest, NsDebug Parameters

- ☐ Run current edited window
- Main window: WINT8

Build Rebuild Check Only Close Help

Segment SEG_NSAGENTTRACE (Librairie NSMisc)

Syntaxe	SEGMENT SEG_NSAGENTTRACE version trEvt module res ctrl id descr ENDSEGMENT			
Champs	version	INTEGER	I	version de vérification du segment (version 1)
	trEvt	INTEGER	I	événement de trace (constantes GTE_*)
	module	CSTRING	O	nom de l'EXE ou de la DLL
	res	CSTRING	O	nom du fichier de ressource NCL/SCR/TPL
	ctrl	CSTRING	O	pour les événements NCL uniquement, "<CLIENT>" pour la fenêtre
	id	CSTRING	O	identifiant (fonction,

				instruction, événement)
	descr	POINTER	I	description du contenu du pointeur d'information (paramètre de trace du callback)

Voir aussi Constantes GTE **, NS TRACE*

Constantes GTE_* (Librairie NSMisc)

Valeurs représentant les événements de trace (GTE = Generator Trace Event).

Syntaxe	Déclaration interne	Description
GTE_ENTERINSTR%	1	début d'une instruction
GTE_LEAVEINSTR%	2	fin d'une instruction
GTE_ENTERFUNC%	3	début d'une fonction
GTE_LEAVEFUNC%	4	fin d'une fonction
GTE_ENTEREVENT%	5	début d'un événement fenêtre
GTE_LEAVEEVENT%	6	fin d'un événement fenêtre

Voir aussi SEG *NSGENTRACE, NS TRACE, MyTrace, SetNSGenTrace*

Gestion de l'ancrage

Constantes NS_AS_* (Librairie NSMisc)

Les constantes NS_AS_* permettent de fixer les contrôles dans une fenêtre.

Syntaxe	Déclaration interne	Description
NS_AS_LEFT	"L"	Ancrage à gauche
NS_AS_RIGHT	"R"	Ancrage à droite
NS_AS_TOP	"T"	Ancrage en haut
NS_AS_BOTTOM	"B"	Ancrage en bas

NS_AS_PLEFT	"%L"	Ancrage proportionnellement à gauche
NS_AS_PRIGHT	"%R"	Ancrage proportionnellement à droite
NS_AS_PTOP	"%T"	Ancrage proportionnellement en haut
NS_AS_PBOTTOM	"%B"	Ancrage proportionnellement en bas
NS_AS_NONE	""	Pas d'ancrage par les côtés, mais par le centre du contrôle, fixe en proportion de la taille de la fenêtre.
NS_AS_BOTTOM_LEFT	"B L"	Ancrage en bas et à gauche du contrôle
NS_AS_TOP_LEFT	"T L"	Ancrage en haut et à gauche du contrôle
NS_AS_BOTTOM_RIGHT	"B R"	Ancrage en bas et à droite du contrôle
NS_AS_TOP_RIGHT	"T R"	Ancrage en haut et à droite du contrôle
NS_AS_WIDTH_TOP	"T L R"	Ancrage en gauche, à droite et en haut
NS_AS_WIDTH_BOTTOM	"B L R"	Ancrage à gauche, à droite et en bas
NS_AS_WIDTH_HEIGHT	"T B L R"	Ancrage en haut, en bas, à gauche et à droite
NS_AS_HEIGHT_LEFT	"T B L"	Ancrage en haut, en bas et à gauche
NS_AS_HEIGHT_RIGHT	"T B R"	Ancrage en haut, en bas et à droite

Ces constantes s'utilisent avec le paramètre dynamique .ANCHOR.

Exemple :

```
local control c
firstcontrol (self%,C)
while miscerror% = 0
  c.ANCHOR = NS_AS_PTOP & NS_AS_PBOTTOM & NS_AS_PLEFT & NS_AS_PRIGHT
  nextcontrol (c)
EndWhile
```

Voir aussi Paramètre dynamique .ANCHOR, ancrage des contrôles (NS-DK ou NatStar).

Tri des tableaux

Instruction NSSORT (Librairie NSMisc)

Permet de trier automatiquement des tableaux.

Syntaxe	NSSORT pdata, Count%, CompFunc, SwapInstr		
Paramètres	pData	POINTER	adresse du tableau de pointeurs
	count%	INTEGER	nombre d'éléments du tableau
	CompFunc	POINTER	pointeur vers la routine de comparaison
	SwapInstr	POINTER	pointeur vers la routine de Swap

1. Le paramètre pData doit être un tableau d'entier ou de pointeurs
2. Le paramètre CompFunc vaut 0 si on désire utiliser la routine de comparaison standard
3. Le paramètre CompSwap vaut 0 si on désire utiliser la routine de swap standard

Prototype de la fonction de comparaison :

```
FUNCTION CompFunc%(POINTER pdata, INTEGER i1%, INTEGER i2%) RETURN INTEGER
```

Prototype de l'instruction de Swap (interversion) :

```
INSTRUCTION SwapInstr POINTER pdata, INTEGER i1%, INTEGER i2%
```

Exemple 1 :

```
;Tri d'un tableau d'entier
const MAX_ITEMS 1000
; array of numbers
global Numbers% [1000]
RANDOMIZE
; fill
for u% = 0 to MAX_ITEMS -1
  numbers%[u%] = random% (10000)
endFor
; sort the array
NSSORT @numbers%, MAX_ITEMS,0,0
```

Exemple 2 :

```
;Tri d'un tableau de segments clients.
segment sCLIENT
  cstring name$ (31)
  cstring firstName$ (31)
  int AGE%
EndSegment

; array of clients
global sCLIENT clients [1000]

; array of pointers used for sort
global pointer pData [1000]

; CompAge%()
; callBack function for age comparison
function CompAge%(pointer @pData[], INTEGER i1%, INTEGER i2%) Return Integer
; return sCLIENT (pdata[i1%]).age% - sCLIENT (pdata[i2%]).age%
endFunction
```

Exemple 3 :

```
local u%
local sCLIENT @c1

; put the clients array in the pointer array
for u% = 0 to MAX_ITEMS-1
  pData[u%] = @clients[u%]
endFor

; call custom routines (CompAge) comparaison of ages
nssort @pdata, MAX_ITEMS, @CompAge%, 0

; Insert array in ListBox
for u% = 0 to MAX_ITEMS -1
  @c1 = sCLIENT (pdata[u%])
  insert at end "Client" && u% && "=" && c1.name$ && "(age=" & c1.age% & ")" to LISTBOX
endFor
```

Autres

Fonction WCCSEND% (Librairie NSMisc)

Permet d'envoyer des messages Windows nécessitant deux paramètres (wParam, lParam) au contrôle Windows 32 implémenté dans NatStar.

Syntaxe	WCCSEND% (Pwcc, Msg, Wparam, LParam)		
Paramètres	Pwcc	POINTER	pointeur sur le contrôle Windows.
	Msg	INTEGER	message Windows à envoyer
	Wparam	INTEGER	spécifie la valeur du paramètre du message
	LParam	INT(4)	spécifie la valeur du paramètre du message
Valeur renvoyée	INT(4)		

Cette fonction est à utiliser lorsque le message Windows indiqué dans la documentation MSDN nécessite deux paramètres.

Exemple 1 :

```
;Le message DTM_SETRANGE du contrôle Date and Time picker contient deux paramètres et doit
donc être utilisé avec WCCSEND%.
;Définition MSDN :
DTM_SETRANGE
wParam = (WPARAM) flags;
lParam = (LPARAM) lpSysTimeArray;
```

Exemple 2 :

```
local Systemtime lsystemtime
local ret%

fill @lsystemtime, sizeof Systemtime, 0
lsystemtime.wyear = 2001
lsystemtime.wmonth = 4
lsystemtime.wdayofweek = 4
lsystemtime.wDay = 12
lsystemtime.wHour = 18
lsystemtime.wMinute = 10
lsystemtime.wSecond = 09
lsystemtime.wMilliseconds = 0
; affecte une valeur à contrôle Date and Time Picker

ret% = WCCSEND%(int DATETIMEPICKER.ctrl,DTM_SETSYSTEMTIME%,GDT_VALID%, @lSystemTime)
```

Fonction WCCSEND0% (Librairie NSMisc)

Permet d'envoyer des messages Windows ne nécessitant aucun paramètre au contrôle Windows 32 implémenté dans NatStar.

Syntaxe	WCCSEND0% (Pwcc, Msg)
---------	-----------------------

Paramètres	Pwcc	POINTER	pointeur sur le contrôle Windows
	Msg	INTEGER	message Windows à envoyer
Valeur renvoyée	INT(4)		

Cette fonction est à utiliser lorsque le message Windows indiqué dans la documentation MSDN ne nécessite aucun paramètre.

Exemple 1 :

```
;Le message DTM_GETMCFONT du contrôle Date and Time picker ne contient aucun paramètre et doit donc être utilisé avec WCCSEND0%.
;Définition MSDN :
DTM_GETMCFONT
wParam = 0;
lParam = 0;
```

Exemple 2 :

```
; lire la valeur d'un contrôle track bar
position%=wccsend0% (int wcctb.ctrl, TBM_GETPOS%)
```

Fonction **WCCSEND1%** (Librairie NSMisc)

Permet d'envoyer des messages Windows nécessitant un paramètre (wParam) au contrôle Windows 32 implémenté dans NatStar.

Syntaxe	WCCSEND1% (<i>Pwcc, Msg, wParam</i>)		
Paramètres	Pwcc	POINTER	pointeur sur le contrôle Windows
	Msg	INTEGER	message Windows à envoyer
	wParam	INTEGER	spécifie la valeur du paramètre du message
Valeur renvoyée	INT(4)		

Cette fonction est donc à utiliser lorsque le message Windows indiqué dans la documentation MSDN nécessite un paramètre.

Exemple 1 :

```
;Le message DTM_GETMCCOLOR du contrôle Date and Time picker un paramètre et doit donc être utilisé avec WCCSEND0%.
;Définition MSDN :
DTM_GETMCCOLOR wParam = (WPARAM)(INT)iColor;
lParam = 0;
```


Exemple 2 :

```
local couleur%
; récupère la couleur de fond d'un contrôle Date Time Picker en code RGB
couleur% = WCCSEND1% (int DATETIMEPICKER.ctrl, DTM_GETMCCOLOR%, MCSC_BACKGROUND%)
```

Fonction WCCSENDPTR% (Librairie NSMisc)

Permet d'envoyer des messages Windows nécessitant un paramètre de type pointeur au contrôle Windows 32 implémenté dans NatStar.

Syntaxe	WCCSENDPTR% (Pwcc, Msg, IParam)		
Paramètres	Pwcc	POINTER	pointeur sur le contrôle Windows
	Msg	INTEGER	message Windows à envoyer
	IParam	INTEGER	spécifie la valeur du paramètre du message
Valeur renvoyée	INT(4)		

Exemple :

```
local Systemtime lsystemtime
local ret%
; retourne la valeur d'un contrôle Date Time Picker

fill @lsystemtime, sizeof Systemtime, 0
ret% = WCCSENDPTR%(int DATETIMEPICKER.ctrl,DTM_GETSYSTEMTIME%,@lSystemTime)
if ret% = GDT_VALID%
efda = lsystemtime.wyear
efdm = lsystemtime.wmonth
efdjs = lsystemtime.wdayofweek
efdj = lsystemtime.wDay
efdh = lsystemtime.wHour
efdm = lsystemtime.wMinute
efds = lsystemtime.wSecond
efdms = lsystemtime.wMilliseconds
endif
```

Fonction WCCSENDSTR% (Librairie NSMisc)

Permet d'envoyer des messages Windows nécessitant un paramètre de type chaîne de caractères au contrôle Windows 32 implémenté dans NatStar.

Syntaxe	WCCSENDSTR% (Pwcc, Msg, IParam)		
Paramètres	Pwcc	POINTER	pointeur sur le contrôle Windows
	Msg	INTEGER	message Windows à envoyer

	IParam	INTEGER	spécifie la valeur du paramètre du message
Valeur renvoyée	INT(4)		

Exemple :

```
;ouvre un fichier .AVI dans un contrôle Animation
local ret%
ret% = WCCSENDSTR%(INT WCCANIM.CTRL, ACM_OPEN%,"c:\temp\f.avi")
```

Fonction **RANDOM%** ([Librairie NSMisc](#))

Retourne un nombre entier aléatoire.

Syntaxe	RANDOM% (<i>entier-maximum</i>)		
Paramètre	entier-maximum	INTEGER	valeur maximum
Valeur retournée	INTEGER		

1. Le nombre entier retourné est compris entre 0 et (entier-maximum - 1)
2. La valeur maximum doit être comprise entre 1 et 32767.
3. **RANDOM%** ne peut être utilisé qu'après avoir initialisé le générateur de nombre aléatoires à l'aide de [RANDOMIZE](#).

Exemple :

```
; Exemple simulant le tirage d'un dé, jusqu'à l'obtention d'un 6. (valeurs possibles: de 1 à 6). Ici, RANDOM%(6) ne peut retourner que des valeurs entières comprises entre 0 et 5.
RANDOMIZE
REPEAT
MOVE RANDOM%(6)+1 TO I%
MESSAGE "Tirage du dé", I%
UNTIL I% = 6
MESSAGE "Gagné !", "Le six a été tiré."
```

Voir aussi [RANDOM#](#), [RANDOMIZE](#)

Fonction **RANDOM#** ([Librairie NSMisc](#))

Retourne un nombre réel aléatoire.

Syntaxe	RANDOM#
Valeur retournée	NUM(8)

1. Le nombre réel retourné est compris entre 0 (inclusif) et 1 (exclusif).

2. RANDOM# ne peut être utilisé qu'après avoir initialisé le générateur de nombres aléatoires à l'aide de RANDOMIZE.

Voir aussi RANDOM%, RANDOMIZE

Instruction RANDOMIZE (Librairie NSMisc)

Initialise le générateur de nombres aléatoires.

Syntaxe	RANDOMIZE
----------------	------------------

L'appel à RANDOMIZE est obligatoire pour pouvoir obtenir un nombre aléatoire avec RANDOM% ou RANDOM#.

Exemple :

Voir exemple illustrant RANDOM%

Voir aussi RANDOM%, RANDOM#

Fonction ASCII2EBDIC\$ (Librairie NSMisc)

Effectue une conversion ASCII vers EBCDIC et retourne l'équivalent EBCDIC de la chaîne ASCII convertie.

Syntaxe	ASCII2EBDIC\$ (chaîne-ASCII)			
Paramètre	chaîne-ASCII	CSTRING		chaîne ASCII à convertir
Valeur retournée	CSTRING			

Voir aussi EBDIC2ASCII\$, *Language NCL: ASC%, CHR\$*

Fonction EBCDIC2ASCII\$ (Librairie NSMisc)

Effectue une conversion EBCDIC vers ASCII et retourne l'équivalent ASCII de la chaîne EBCDIC convertie.

Syntaxe	EBCDIC2ASCII\$ (EBCDIC-string)			
Paramètre	EBCDIC-string	CSTRING		caractère EBCDIC à convertir
Valeur retournée	CSTRING			

Voir aussi ASCII2EBDIC\$, *Language NCL: ASC%, CHR\$*

Fonction STRING\$ (Librairie NSMisc)

Formate un nombre réel et retourne une chaîne résultante.

Syntaxe	STRING\$ (réel, chaîne-format)		
Paramètres	réel	NUM(8)	réel à formater
	chaîne-format	CSTRING	format à appliquer
Valeur retournée	CSTRING		

Le format défini dans chaîne-format est toujours exprimé selon le format américain ("," pour séparateur décimal, "." pour séparateur des milliers). Par contre, la chaîne retournée par STRING\$ dépend des séparateurs définis dans le Panneau de Configuration et modifiables par les instructions [SETDECIMALSEPARATOR](#) et [SETTHOUSANDSSEPARATOR](#). Cela assure une complète portabilité des applications lors de l'utilisation de STRING\$ quels que soient les séparateurs définis dans le Panneau de Configuration : seul le résultat affiché dépend de la machine sur laquelle fonctionne l'application.

1. Les formats acceptés dans chaîne-format sont décrits dans l'annexe A de ce manuel.
2. Utilisez la fonction [ESTRING\\$](#) si le réel à formater est de type NUM(10).
3. Les fonctions [DATE\\$](#) et [TIME\\$](#) de la librairie NSDate permettent aussi de formater des dates et heures.

Voir les [Formats d'affichages](#)

Exemple :

```
; Dans cet exemple (voir commentaire 1), le séparateur des milliers est supposé être l'espace, et le séparateur décimal est supposé être la virgule.
S$ = STRING$(1234.567, "0"); S$ vaut "1235"
S$ = STRING$(1234.567, "0.0"); S$ vaut "1234,6"
S$ = STRING$(1234.567, "# ##0.00"); S$ vaut "1 234,57"
S$ = STRING$(1234.567, "0.0E+00"); S$ vaut "1,2E+03"
S$ = STRING$(0.12345, "0%"); S$ vaut "12%"

S$ = STRING$(0, "yyyy-m-d") ; S$ vaut "1900-1-1"
S$ = STRING$(0, "hh:mm:ss") ; S$ vaut "00:00:00"
```

Voir aussi [ESTRING\\$](#), [SETDECIMALSEPARATOR](#), [SETTHOUSANDSSEPARATOR](#), [CURRENCY\\$](#), [DATE\\$](#), [TIME\\$](#) (librairie NSDate)

Fonction [ESTRING\\$](#) (Librairie NSMisc)

Formate un nombre réel et retourne une chaîne résultante.

Syntaxe	ESTRING\$ (réel, chaîne-format)
----------------	--

Paramètres	réel	NUM(10)		réel à formater
	chaîne-format	CSTRING		format à appliquer
Valeur retournée	CSTRING			

Voir les commentaires de la fonction STRING\$. La seule différence avec cette dernière est que le nombre à formater est de type NUM(10).

Voir aussi STRING\$, SETDECIMALSEPARATOR, SETTHOUSANDSSEPARATOR, CURRENCY\$, DATE\$, TIME\$ (librairie NSDATE)

Fonction CURRENCY\$ (Librairie NSMisc)

Formate un nombre réel selon les définitions monétaires définies dans le système d'exploitation et retourne la chaîne résultante.

Syntaxe	CURRENCY\$ (number)			
Paramètre	number	NUM(8)		nombre à formater
Valeur retournée	CSTRING			

1. Le format appliqué est celui défini dans le Panneau de Configuration, éventuellement modifié avec les séparateurs précisés par SETDECIMALSEPARATOR et SETTHOUSANDSSEPARATOR si ces instructions ont été préalablement appelées.
2. Utilisez la fonction ECURRENCY\$ si le type du nombre réel que vous souhaitez formater est NUM(10).

Exemple :

```
; Si "F" est le symbole monétaire défini en suffixe dans le Control Panel, alors :
SETDECIMALSEPARATOR ","
SETTHOUSANDSSEPARATOR " "
MOVE CURRENCY$(2371.4) TO S$ ; S$ contient "2 371,40 €"
```

Remarque : sa précision peut être améliorée grâce à l'utilisation du Flag RoundCurrency dans le fichier NSLIB.INI

Voir aussi ECURRENCY\$, SETDECIMALSEPARATOR, SETTHOUSANDSSEPARATOR, STRING\$

Fonction ECURRENCY\$ (Librairie NSMisc)

Formate un nombre réel selon les définitions monétaires définies dans le système d'exploitation et retourne la chaîne résultante.

Syntaxe	ECURRENCY\$ (<i>nombre</i>)		
Paramètre	nombre	NUM(10)	nombre à formater
Valeur retournée	CSTRING		

Voir les commentaires de la fonction CURRENCY\$. La seule différence avec cette dernière est que le nombre à formater est de type NUM(10).

Voir aussi CURRENCY\$, SETDECIMALSEPARATOR, SETTHOUSANDSSEPARATOR, STRING\$

Constantes MB_ *% (Librairie NSMisc)

Valeurs représentant les éléments d'une boîte de message.

Syntaxe	Déclaration interne	Description
MB_ABORT%	4	Bouton Abort (Abandon)
MB_APPLICATION%	32768	spécifie que la boîte de message est modale pour l'application : seule l'application qui fait le MESSAGE% est bloquée tant que la boîte n'est pas fermée.
MB_CANCEL%	2	Bouton Cancel (Annulation)
MB_DEFBUTTON1%	131072	Premier bouton sélectionné par défaut
MB_DEFBUTTON2%	262144	Second bouton sélectionné par défaut
MB_DEFBUTTON3%	524288	Troisième bouton sélectionné par défaut
MB_ENTER%	1	Bouton Enter (Entrée)

MB_ERROR%	256	valeur retournée par MESSAGE% si la fonction a échoué. Une erreur est donc survenue pendant le traitement de MESSAGE% dans des circonstances extrêmes comme, par exemple, plus assez de mémoire disponible pour afficher la boîte.
MB_ICONASTERISK%	1024	"i" d'information (équivalent à SPTR_ICONINFORMATION%)
MB_ICONEXCLAMATION%	2048	Point d'exclamation (équivalent à SPTR_ICONWARNING%)
MB_ICONHAND%	4096	Panneau Stop (équivalent à SPTR_ICONERROR%)
MB_ICONQUESTION%	8192	Point d'interrogation (équivalent à SPTR_ICONQUESTION%)
MB_IGNORE%	8	Bouton Ignore (Ignorer)
MB_MOVEABLE%	65536	fait apparaître la boîte de message avec une barre de titre (et un menu système) permettant de la déplacer. Par défaut, si MB_MOVEABLE% n'est pas spécifié, la boîte ne peut être déplacée.
MB_NO%	128	Bouton No (Non)
MB_OK%	16	Bouton OK (Validation)
MB_RETRY%	32	Bouton Retry (Reprise)

MB_SYSTEM% (obsolète)	16384	spécifie que la boîte de message est modale pour le système : toutes les applications sont bloquées tant que la boîte n'est pas fermée.
MB_YES%	64	Bouton Yes (Oui)

1. Ces constantes sont utilisées par le dernier paramètre ainsi que par le code retour de la fonction MESSAGE%
2. La langue dans laquelle le texte de ces boutons est affiché dépend de la langue du système utilisé.
3. Il est impossible de combiner plus de trois Push-Buttons MB_*. De plus, toutes les combinaisons de 1, 2 ou 3 boutons ne sont pas implémentés. Voici les combinaisons supportées de Push-Buttons :

Premier	Second	Troisième
MB_OK%		
MB_OK%	MB_CANCEL%	
MB_ENTER%		
MB_ENTER%	MB_CANCEL%	
MB_RETRY%	MB_CANCEL%	
MB_YES%	MB_NO%	
MB_YES%	MB_NO%	MB_CANCEL%
MB_ABORT%	MB_RETRY%	MB_IGNORE%

4. Il ne peut y avoir qu'un seul bouton sélectionné par défaut au démarrage de la boîte de message. Si aucun MB_DEFBUTTON*% n'est spécifié, le premier bouton est sélectionné. L'ordre d'un bouton ne dépend pas de l'ordre utilisé lors de l'addition des constantes push-buttons MB_*% (car A+B est égal à B+A), mais dépend de la colonne de MB_*% de la liste des combinaisons supportées de Push-Buttons. Ainsi le "second" bouton de MB_CANCEL% + MB_ENTER% est MB_CANCEL%.
5. L'icône affichée dans la boîte de message est spécifiée à l'aide d'une des constantes MB_ICON*%. Il ne peut y avoir qu'une seule icône dans la boîte de message. Si aucun MB_ICON*% n'est spécifié, la boîte apparaît sans icône.
6. Le comportement de la boîte de message peut être précisé à l'aide des constantes MB_SYSTEM%, MB_APPLICATION% (si aucun de ces deux

comportements n'est spécifié, la boîte est modale pour l'application) et MB_MOVEABLE%.

7. MB_ERROR% est la seule constante MB_+% à ne pas être employable comme paramètre de MESSAGE%.

Voir aussi MESSAGE%

Fonction MESSAGE% (Librairie NSMisc)

Ouvre une boîte de message composée de push-buttons et optionnellement d'une icône. La fonction retourne la valeur du push-button enfoncé.

Syntaxe	MESSAGE% (handle-fenêtre, chaîne-titre, chaîne-message, item-aide, style-boîte)			
Paramètres	handle-fenêtre	POINTER		handle de la fenêtre origine du message
	chaîne-titre	CSTRING		titre de la boîte
	chaîne-message	CSTRING		message affiché dans la boîte
	item-aide	INTEGER		item d'aide appelé si l'utilisateur appuie sur le bouton Help.
	style-boîte	INT(4)		style de la boîte
Valeur retournée	INT(4) L'une des constantes MB_+%			

1. handle-fenêtre est généralement le handle de la fenêtre qui fait le MESSAGE%, c'est-à-dire SELF%.

2. Le paramètre item-aide n'est pas opérationnel. De fait, la constante MB_HELP% est obsolète.

3. style-boîte est une combinaison de constantes MB_+% permettant de préciser les push-buttons affichés, et l'icône éventuelle. Si MB_MOVEABLE% y est également spécifié, une barre de titre et un menu système sont rajoutés à la fenêtre. Dans ce cas, lorsque la boîte est fermée par son menu système (double-clic, [Alt]+[F4], ou sélection de l'élément Close), la fonction MESSAGE% retourne MB_CANCEL%.

Exemple :

```
IF MESSAGE(SELF%, "Titre", "Ceci est un message", 0, MB_RETRY% + MB_CANCEL% +
MB_ICONHAND% + MB_MOVEABLE%) = MB_RETRY%
MESSAGE "Boîte MESSAGE% fermée", "par appui du bouton Retry"
```

```
ELSE
MESSAGE "Boîte MESSAGE% fermée", "par appui du bouton Cancel ou par son menu système"
ENDIF
```



Voir aussi Constantes MB *, Librairie NSHELP, Language NCL : MESSAGE , ASK2%, ASK3%

Fonction GETDECIMALSEPARATOR\$ (Librairie NSMisc)

Retourne le caractère employé pour le séparateur décimal au sein de l'application.

Syntaxe	GETDECIMALSEPARATOR\$
Valeur retournée	CSTRING

S'il n'y a jamais eu de SETDECIMALSEPARATOR auparavant, cette fonction retourne le séparateur par défaut précisé dans le Panneau de Configuration.

Voir aussi STRING\$, SETDECIMALSEPARATOR, GETTHOUSANDSSEPARATOR\$, SETTHOUSANDSSEPARATOR

Fonction GETTHOUSANDSSEPARATOR\$ (Librairie NSMisc)

Retourne le caractère employé pour le séparateur des milliers au sein de l'application.

Syntaxe	GETTHOUSANDSSEPARATOR\$
Valeur retournée	CSTRING

S'il n'y a jamais eu de SETTHOUSANDSSEPARATOR auparavant, cette fonction retourne le séparateur par défaut précisé dans le Panneau de Configuration.

Voir aussi STRING\$, SETTHOUSANDSSEPARATOR, GETDECIMALSEPARATOR\$, SETDECIMALSEPARATOR

Instruction SETDECIMALSEPARATOR (Librairie NSMisc)

Définit le caractère employé pour le séparateur décimal.

Syntaxe	SETDECIMALSEPARATOR <i>string</i>		
Paramètre	string	CSTRING	séparateur

1. Ce séparateur est uniquement pris en compte au sein de l'application : cela ne modifie pas le séparateur par défaut précisé dans le Panneau de Configuration et qui reste valable pour toutes les autres applications n'ayant pas spécifié de séparateur particulier.
2. Le séparateur décimal ainsi défini est utilisé par les fonctions STRING\$ et CURRENCY\$ ainsi que par les contrôles Entry-Fields et CBE au format "Number".
3. SETDECIMALSEPARATOR "!" permet d'utiliser le séparateur précisé dans le Panneau de Configuration.
4. Le paramètre NsWPForceNatNumberSep du fichier NSLIB.INI permet de modifier le séparateur de décimales et de milliers dans NS-DK et NSWP.

Exemple 1 :

```
SETDECIMALSEPARATOR "." ; à l'anglaise
MOVE STRING$ (0.2, "0.0") TO S$ ; S$ vaut "0.2"
```

Exemple 2 :

```
SETDECIMALSEPARATOR "," ; à la française
MOVE STRING$ (0.2, "0,0") TO S$ ; S$ vaut "0,2"
```

Voir aussi STRING\$, CURRENCY\$, GETDECIMALSEPARATOR\$, GETTHOUSANDSSEPARATOR\$, SETTHOUSANDSSEPARATOR

Instruction SETTHOUSANDSSEPARATOR (Librairie NSMisc)

Définit le caractère employé pour le séparateur des milliers.

Syntaxe	SETTHOUSANDSSEPARATOR <i>string</i>		
Paramètre	string	CSTRING	I séparateur

1. Ce séparateur est uniquement pris en compte au sein de l'application : cela ne modifie pas le séparateur par défaut précisé dans le Panneau de Configuration et qui reste valable pour toutes les autres applications n'ayant pas spécifié de séparateur particulier.
2. Le séparateur des milliers ainsi défini est utilisé par les fonctions STRING\$ et CURRENCY\$ ainsi que par les contrôles Entry-Fields et CBE au format "Number".
3. Avec STRING\$, le séparateur des milliers doit être indiqué dans la chaîne de format passée à STRING\$ pour être pris en compte. La chaîne de format utilise la virgule (,) pour indiquer le séparateur des milliers, et la virgule se réfère au séparateur défini par SETTHOUSANDSSEPARATOR.
4. SETTHOUSANDSSEPARATOR "!" permet d'utiliser le séparateur précisé dans le Panneau de Configuration.

5. Le paramètre `NSWPForceNatNumberSep` du fichier `NSLIB.INI` permet de modifier le séparateur de décimales et de milliers dans `NS-DK` et `NSWP`.

Exemple 1 :

```
SETTHOUSANDSSEPARATOR " " ; à l'ancienne (espace séparateur des milliers)
MOVE STRING$ (2300, " #,##0") TO S$ ; S$ vaut "2 300"
```

Exemple 2 :

```
SETTHOUSANDSSEPARATOR "" ; absence de séparateur
MOVE STRING$ (2300, "###0") TO S$ ; S$ vaut "2300"
```

Voir aussi [STRING\\$](#), [CURRENCY\\$](#), [GETTHOUSANDSSEPARATOR\\$](#), [GETDECIMALSEPARATOR\\$](#), [SETDECIMALSEPARATOR](#)

Fonction **COMPILEGREGEXPR%** (Librairie **NSMisc**)

Valide un masque d'expression, le compile et le stocke dans une forme compactée dans un buffer.

Syntaxe	COMPILEGREGEXPR% (<i>masque-expression, adresse-buffer, taille-buffer</i>)			
Paramètres	expression-mask	CSTRING	I	expression à vérifier
	buffer-address	POINTER	I	adresse du buffer de stockage
	buffer-size	INTEGER	I	taille du buffer
Valeur retournée	INT(1) 0 : Aucune erreur 1 : Dépassement de la taille du buffer 2 : Erreur de syntaxe détectée			

1. Le masque d'expression s'applique à une chaîne caractère par caractère.
2. Syntaxe de description des expressions régulières :
 - a) [-] désigne un ensemble de caractères autorisés.

exemple :

[A-Z] autorise les majuscules de A à Z.

[a-k] autorise les minuscules de a à k.

[A-JS-Ws-w] autorise les majuscules de A à Z et de S à W ainsi que les minuscules de s à w.

- b) Une lettre spécifiée sans rien d'autre indique que seule cette lettre est autorisée.

exemple :

Madame n'autorise que la saisie de Madame.

Par contre, [Mm]onsieur autorise la saisie de Monsieur et de monsieur

c) ^ autorise tous les caractères SAUF ceux qui suivent le ^.

exemple :

[^A-Z0] autorise tout sauf A à Z et le zéro.

[^EJMP] autorise tout sauf E J M et P.

d) % autorise 0 à n caractères quelconques.

exemple :

%Monsieur% autorise toute phrase comprenant Monsieur, ainsi "Cher Monsieur Deschamps" est correct.

e) _ autorise un caractère quelconque.

exemple :

00_99 autorise tout mot de 6 lettres encadré par 00 et 99 (002499, 00az99, etc...).

f) * indique que ce qui précède peut être répété de 0 à n fois.

exemple :

[A-J0-3]* autorise de 0 à n caractères entre A et J et 0 et 3.

[^EL246]* autorise de 0 à n caractères différents de E,L,2,4 ou 6.

g) \ sert de caractère d'échappement.

Il doit être utilisé chaque fois qu'un caractère utilisé pour la syntaxe d'une expression doit être interprété comme un caractère 'normal'.

Exemple :

C:\\[A-Z]* pour autoriser les chaînes commençant par C:\\, suivies d'une lettre majuscule et se terminant par *.

Exemple :

```
SEGMENT BUFFER
CHAR DATA[300]
ENDSEGMENT
LOCAL BUF%
NEW BUFFER,BUF%

IF COMPILEREGREXPR('%[Mm]onsieur%',BUF%,300)=0
...
IF EXECUTEREGREXPR('Cher Monsieur Desroches',BUF%)
...;traitement chaîne correcte
ELSE
...;traitement chaîne incorrecte
ENDIF
ELSE
...;traitement erreur
ENDIF
DISPOSE BUF%
```

Voir aussi **EXECUTEREGREXPR%**

Fonction EXECUTEREGREXPR% (Librairie NSMisc)

Teste si une chaîne de caractères respecte une expression régulière.

Syntaxe	EXECUTEREGREXPR% (chaîne, adresse-buffer)		
Paramètres	chaîne	CSTRING	chaîne à tester
	adresse-buffer	POINTER	adresse du buffer où a été rangée l'expression régulière compilée.
Valeur retournée	INT(1) TRUE% : La chaîne respecte l'expression régulière. FALSE% : La chaîne ne respecte pas l'expression régulière.		

L'expression régulière doit avoir été compilée auparavant avec COMPILEREGREXPR% et sauvegardée dans le buffer d'adresse adresse-buffer.

Exemple :

```
SEGMENT BUFFER
CHAR DATA[300]
ENDSEGMENT
LOCAL BUF%
NEW BUFFER,BUF%

IF COMPILEREGREXPR%('[Mm]onsieur',BUF%,300)=0
...
IF EXECUTEREGREXPR%('Cher Monsieur Desroches',BUF%)
...;traitement chaîne correcte
ELSE
...;traitement chaîne incorrecte
ENDIF
ELSE
...;traitement erreur
ENDIF
DISPOSE BUF%
```

Voir aussi COMPILEREGREXPR%

Fonction GETBITMAPCONTROLSDEFAULTS% (Librairie NSMisc)

Permet de connaître le fonctionnement par défaut des contrôles bitmaps.

Syntaxe	GETBITMAPCONTROLSDEFAULTS%
---------	----------------------------

Valeur retournée	INTEGER <u>Constante DEFBMP*</u>
-------------------------	-------------------------------------

Les valeurs retournées sont une combinaison (somme) des flags suivants :

1. DEFBMPTRANSPARENT% (1)

Le contrôle bitmap utilise le pixel à l'extrémité en haut à gauche comme couleur transparente.

2. DEFBMPBUTTONIZED% (2)

Le contrôle bitmap a le dessin automatique des bordures et la simulation des états Pressed et Disabled des contrôles bitmap de type PushButton n'ayant qu'une bitmap (Released).

Voir aussi SETBITMAPCONTROLSDEFAULTS

Instruction SETBITMAPCONTROLSDEFAULTS (Librairie NSMisc)

Permet de modifier le fonctionnement par défaut des contrôles bitmaps.

Syntaxe	SETBITMAPCONTROLSDEFAULTS defaults%		
Paramètre	defaults%	INT	Constante <u>DEFBMP*</u>

La transparence d'une image .JPG est fortement déconseillée.

Voir aussi GETBITMAPCONTROLSDEFAULTS%, Constantes TRANSP *%, Constantes DEFBMP*

Constantes BMPF_ *% (Librairie NSMisc)

Constantes utilisées pour le paramétrage dynamique .RELIEF d'un bouton Bitmap.

Syntaxe	Déclaration interne	Description
BMPF_DEFBTNBORDER%	0	utilisation du mode par défaut
BMPF_NOBTNBORDER%	1	ne dessine pas automatiquement les bordures et les états Pressed/disabled
BMPF_HAVEBTNBORDER%	2	dessine toujours automatiquement les bordures et les états Pressed/disabled
BMPF_BTNBORDERMASK%	3	masque où définir/extraire le comportement du bouton Bitmap

Voir aussi Constantes DEFBMP*%, GETBITMAPCONTROLSDEFAULTS%, SETBITMAPCONTROLSDEFAULTS

Constantes DEFBMP*% (Librairie NSMisc)

Valeurs indiquant le fonctionnement des contrôles bitmaps.

Syntaxe	Déclaration interne	Description
DEFBMPTRANSPARENT%		Le contrôle bitmap utilise le pixel à l'extrémité en haut à gauche comme couleur transparente.
DEFBMPBUTTONIZED%		Le contrôle bitmap a automatiquement le dessin des bordures et la simulation des états Pressed et Disabled pour les contrôles bitmap de type PushButton n'ayant qu'une bitmap (Released).

La transparence d'une image .JPG est fortement déconseillée.

Voir aussi [GETBITMAPCONTROLSDEFAULTS%](#), [SETBITMAPCONTROLSDEFAULTS](#), [Constantes TRANSP *%](#)

Constantes TRANSP_*% (Librairie NSMisc)

Symbolisent les différentes valeurs de la couleur de transparence des bitmaps.
Commentaires

Syntaxe	Déclaration interne	Description
TRANSP_NEVER%	-6	la bitmap est toujours opaque
TRANSP_DEFAULT%	-5	utilise la couleur par défaut définie pour le projet (en général, le pixel le plus en haut à gauche)
TRANSP_BOTRIGHT%	-4	utilise le coin inférieur droit
TRANSP_TOPRIGHT%	-3	utilise le coin supérieur droit
TRANSP_BOTLEFT%	-2	utilise le coin inférieur gauche
TRANSP_TOPLEFT%	-1	utilise le coin supérieur gauche
valeur entre 0 et 15		la couleur correspondante est utilisée comme couleur de transparence

Voir aussi [STV_GETTRANSPARENCY%](#), [STV_SETTRANSPARENCY](#), [GPI_TRANSPARENTBITBLT](#)

Fonction SHAREDMEMALLOCERROR% (Librairie NSMisc)

Détermine si la mémoire partagée nommée a déjà été allouée ou non (par exemple par une autre tâche).

Syntaxe	SHAREDMEMALLOCERROR%
Valeur retournée	INT(4) 0 si l'allocation a bien eu lieu 1 si elle n'a pas eu lieu (mémoire partagée nommée déjà allouée ou erreur d'allocation mémoire).

Exemple :

```

NEW SEGSTR, SHMEMPTR%, 'SHARED'
IF SHMEMPTR% = 0
; Un pointeur nul a été retourné
MESSAGE 'Erreur à l'allocation', 'Une erreur est survenue à l'utilisation de NEW'
ELSE
; Un pointeur a été retourné, mais est ce que la zone mémoire a déjà été allouée ?

IF SHAREDMEMALLOCERROR% <> 0
; La mémoire était déjà allouée pas besoin de l'initialiser
ELSE
; La mémoire n'était pas déjà allouée, alors on peut en profiter pour l'initialiser ici
...
ENDIF
ENDIF

```

Fonction TEST_MEMORY_ACCESS% (Librairie NSMisc)

Test si un bloc de mémoire est accessible en lecture et/ou en écriture. Les paramètres sont le pointeur sur le bloc, sa longueur en octets et le mode d'accès à tester (constantes TEST_READ_ACCESS = 1 et TEST_READWRITE_ACCESS = 2).

Syntaxe	TEST_MEMORY_ACCESS% (<i>adr,size,mode</i>)		
Paramètres	adr	POINTER	adresse d'un buffer contenant les données à convertir
	size	INT(4)	taille du buffer, en octets

	mode	INTEGER	I	Mode d'accès à tester
Valeur retournée	INT(1)			TRUE% si la mémoire est accessible, et FALSE% sinon

Dans le mode d'accès à tester, utiliser les constantes TEST_READ_ACCESS et TEST_READWRITE_ACCESS.

Voir aussi Constantes TEST_READ*

Constantes TEST_READ* (Librairie NSMisc)

Valeurs représentant le mode d'accès à tester dans la fonction TEST_MEMORY_ACCESS%.

Syntaxe	Déclaration interne	Description
TEST_READ_ACCESS	1	Mode d'accès en lecture
TEST_READWRITE_ACCESS	2	Mode d'accès en lecture/écriture

Voir aussi TEST_MEMORY_ACCESS%

Fonction NSMEMCOMP% (Librairie NSMisc)

Compare deux blocs de mémoire.

Syntaxe	NSMEMCOMP% (pMem1, pMem2, size%)			
Paramètres	pMem1	POINTER	I	pointeur sur un bloc de mémoire
	pMem2	POINTER	I	pointeur sur un bloc de mémoire
	size%	INTEGER	I	nombre d'octets à comparer
Valeur retournée	INT(1) Retourne 0 si les deux blocs de mémoire sont identiques, autrement retourne la différence entre les deux octets.			

Voir aussi NSMEMPOS%

Constantes CHS_ *% (Librairie NSMisc)

Constantes identifiant les jeux de caractères reconnus.

Syntaxe	Déclaration interne	Description
CHS_ASCII%	0	jeu de caractères ASCII (PC)
CHS_ASCIIIMAC%	1	jeu de caractères ASCII (Macintosh)
CHS_ANSI%	2	jeu de caractères ANSI
CHS_EBCDIC%	3	jeu de caractères EBCDIC
CHS_LATIN1	4	jeu de caractères ISO 8859-1
CHS_ISO8859_1	5	jeu de caractères ISO 8859-1
CHS_ISO8859_15	17	jeu de caractères ISO 8859-15
CHS_PC437	7	jeu de caractères IBM Ascii graphical
CHS_HPROMAN8	8	jeu de caractères HP Roman 8
CHS_ROMAN	9	jeu de caractères Macintosh character set
CHS_EBCDIC_037	10	jeu de caractères IBM EBCDIC American
CHS_EBCDIC_297	11	jeu de caractères IBM EBCDIC French
CHS_EBCDIC_500	12	jeu de caractères IBM EBCDIC International
CHS_EBCDIC_273	13	jeu de caractères IBM EBCDIC German
CHS_EBCDIC_284	14	jeu de caractères IBM EBCDIC Spanish
CHS_EBCDIC_280	15	jeu de caractères IBM EBCDIC Italian
CHS_EBCDIC_285	16	jeu de caractères IBM EBCDIC UK
CHS_UTF8	255	jeu de caractères UTF-8 Unicode

Voir aussi GETHOSTCHARSET%, GETCURRENTCHARSET%, TRANSLATECHARS

Fonction GETCURRENTCHARSET% (Librairie NSMisc)

Retourne le jeu de caractères utilisé par l'application.

Syntaxe	GETCURRENTCHARSET%
----------------	---------------------------

Valeur retournée	INTEGER Une des constantes <u>CHS</u> *%
-------------------------	---

Cette fonction est utile lors du développement d'une application multi-target devant fonctionner sur différents systèmes.

Exemple :

```
LOCAL S$(100)
MOVE ENTRY TO S$
; Conversion de la chaîne en ASCII si jeu de caractères courant en ANSI
IF GETCURRENTCHARSET% = CHS_ANSI%
TRANSLATECHARS @S$, length S$, CHS_ANSI%, CHS_ASCII%
ENDIF
```

Voir aussi GETHOSTCHARSET%, TRANSLATECHARS , TRANSLATECHARS2% TRANSLATECHS\$ SETCURRENTCHARSET et constantes CHS *%

Instruction SETCURRENTCHARSET (Librairie NSMisc)

Permet de changer la page de code (charset) active.

Syntaxe	SETCURRENTCHARSET chs%		
Paramètre	chs%	INTEGER	I Constante <u>CHS</u> *%

À utiliser avec précaution et uniquement si aucune fenêtre de l'application n'est ouverte.

Voir aussi Constantes CHS *%, GETCURRENTCHARSET%, GETHOSTCHARSET%, TRANSLATECHARS, TRANSLATECHARS2%

Fonction GETHOSTCHARSET% (Librairie NSMisc)

Retourne le jeu de caractères utilisé par le système de la machine où fonctionne l'application.

Syntaxe	GETHOSTCHARSET%
Valeur retournée	INTEGER L'une des constantes <u>CHS</u> *%

Cette fonction est utile lors du développement d'une application multi-target devant fonctionner sur différents systèmes.

Voir aussi GETCURRENTCHARSET%, TRANSLATECHARS , constantes CHS *%, TRANSLATECHARS2%, TRANSLATECHS\$ SETCURRENTCHARSET

Instruction TRANSLATECHARS (Librairie NSMisc)

Convertit un buffer de taille donnée d'un jeu de caractères dans un autre.

Syntaxe	TRANSLATECHARS <i>adresse-buffer, taille-buffer, jeu-source, jeu-cible</i>		
Paramètres	adresse-buffer	POINTER	adresse d'un buffer contenant les données à convertir
	taille-buffer	INTEGER	taille du buffer, en octets
	jeu-source	INTEGER	jeu de caractères source
	jeu-cible	INTEGER	jeu de caractères cible

1. Les jeux de caractères doivent être spécifiés à l'aide des constantes CHS *%.
2. Cette instruction est utile lors du développement d'une application multi-target devant fonctionner à la fois sur des machines Unix/Linux et Windows.

Exemple :

```
LOCAL S$(100)
MOVE ENTRY TO S$
; Conversion de la chaîne en ASCII si jeu de caractères courant en ANSI
IF GETCURRENTCHARSET% = CHS_ANSI%
TRANSLATECHARS @S$, length S$, CHS_ANSI%, CHS_ASCII%
ENDIF
```

Voir aussi TRANSLATECHARS2%, TRANSLATECHS\$ GETHOSTCHARSET%, GETCURRENTCHARSET%, SETCURRENTCHARSET constantes CHS *%

Fonction **TRANSLATECHARS2%** (Librairie NSMisc)

Convertit le jeu de caractères d'un buffer d'entrée dans un buffer de sortie et dans un jeu de caractères différent.

Syntaxe	TRANSLATECHARS2% (<i>buf_in, len_in%, chs_in%, buf_out, size_out%, chs_out%</i>)		
Paramètres	buf_in	POINTER	Adresse d'un

			buffer contenant les données à convertir
	len_in%	INTEGER	Longueur du buffer d'entrée
	chs_in%	INTEGER	Jeu de caractères source
	buf_out	POINTER	Adresse du buffer de sortie
	size_out%	INTEGER	Taille maximale en octets du buffer de sortie
	chs_out%	INTEGER	Jeu de caractères cible
Valeur retournée	INTEGER		

1. Les jeux de caractères doivent être spécifiés à l'aide des constantes CHS_.*%.
2. La valeur retournée correspond au nombre d'octets en sortie.
3. Cette fonction est à utiliser obligatoirement à la place de TRANSLATECHARS lorsque l'une des deux pages de code est CHS_UTF8. En effet, dans ce cas, la chaîne en entrée n'a pas toujours la même taille que celle en sortie (ce que TRANSLATECHARS ne permet pas).
4. Pour que le caractère de terminaison "0 binaire" de la string de destination soit positionné, il faut que le paramètre len_in% soit égal à la longueur de la string d'entrée + 1. Sinon la chaîne de destination risque de ne pas avoir de caractère de terminaison.

Voir aussi TRANSLATECHARS, TRANSLATECHS\$ GETHOSTCHARSET%, GETCURRENTCHARSET%, SETCURRENTCHARSET, **constantes CHS_.*%**

Fonction TRANSLATECHS\$ (Librairie NSMisc)

Convertit la chaîne de caractères C dans un jeu de caractères différent.

Syntaxe	TRANSLATECHS% (S\$, chs_in%, chs_out%)		
Paramètres	S\$	CSTRING	Chaîne de caractères C
	chs_in%	INTEGER	Jeu de caractères source
	chs_out%	INTEGER	Jeu de caractères cible
Valeur retournée	CSTRING		

1. Les jeux de caractères doivent être spécifiés à l'aide des constantes CHS *%.
2. La chaîne C est limitée à 255 caractères.

Voir aussi TRANSLATECHARS2%, TRANSLATECHARS, GETHOSTCHARSET%, GETCURRENTCHARSET%, SETCURRENTCHARSET, Constantes CHS *%

Fonction GETVIRTUALKEYSTATE% (Librairie NSMisc)

Retourne un booléen indiquant si la touche code (VK_*%) est enfoncée (TRUE%) ou relâchée (FALSE%).

Syntaxe	GETVIRTUALKEYSTATE% (vkey%)		
Paramètre	vkey%	INTEGER	touche code
Valeur retournée	INT(1)		

Instruction CHANGEWINDOWSTYLE (Librairie NSMisc)

Change dynamiquement les attributs de style des fenêtres. Le paramètre removestyles% est une combinaison de styles à retirer de la fenêtre. Le paramètre addstyles% est une combinaison de styles à ajouter à la fenêtre.

Syntaxe	CHANGEWINDOWSTYLE win, removestyles%, addstyles%
---------	--

Paramètres	win%	POINTER	handle de la fenêtre dont on veut changer les attributs de style.
	removestyles%	INT(4)	combinaison de constantes <u>WS_*</u> de styles à retirer de la fenêtre.
	addstyles%	INT(4)	combinaison de constantes <u>WS_*</u> de styles à ajouter à la fenêtre.

Exemples :

```
; Enlève la barre de titre à la fenêtre et lui rajoute une barre de défilement verticale
et une barre de défilement horizontale.
ChangeWindowStyle hWnd%, WS_TITLE%, WS_HORZSCROLLBAR% bor WS_VERTSCROLLBAR%

; passe la bordure d'une fenêtre de fine à retaillable
ChangeWindowStyle hWnd%, WS_BORDER%, WS_SIZEBORDER%
```

Voir aussi Constantes WS_*

Constantes WS_* (Librairie NSMisc)

Les constantes WS_* sont utilisées conjointement avec l'instruction CHANGIEWINDOWSTYLE pour changer dynamiquement le style des fenêtres.

Syntaxe	Déclaration interne	Description
WS_TITLE%	\$00000001	Montre ou cache la barre de titre.
WS_BORDER%	\$00000002	Positionne la bordure fine.
WS_SIZEBORDER%	\$00000004	Positionne la bordure « retaillable ».

WS_DLGBORDER%	\$00000008	Positionne la bordure « boîte de dialogue ».
WS_VERTSCROLLBAR	\$00000010	Montre ou cache la barre verticale de défilement.
WS_HORIZSCROLLBAR%	\$00000020	Montre ou cache la barre horizontale de défilement
WS_NOBYTEALIGN%	\$00002000	Doit être présent pour que la position horizontale ne soit pas un multiple de 8 pixels (constante à utiliser uniquement en ajout, on ne peut pas la retirer).
WS_QUIT%	\$00008000	Positionne ou enlève l'option « Quit on Close ».
WS_MINIMIZE%	\$00010000	Montre le bouton Minimize.
WS_SYSMENU%	\$04000000	Montre ou cache le menu système
WS_MINMAX%	\$08000000	Montre ou cache les boutons Minimize/Maximize.

Les autres constantes ne sont pas prises en compte. WS_OVER%, WS_BELOW%, WS_DEFAULTICON%, WS_SHELLPOSITION%, WS_SIZEREDRAW%, WS_TASKLIST%, WS_SAVEBITS%, WS_MENUBAR%, WS_SECONDARY%, WS_MDI%

Voir aussi CHANGEWINDOWSTYLE

Fonction DLG_CHECKCONTROLFOCUS% (Librairie NSMisc)

Vérifie que le Custom Control spécifié peut prendre le focus.

Syntaxe	DLG_CHECKCONTROLFOCUS% (ctrl)		
Paramètre	ctrl	CONTROL	nom du Custom Control
Valeur retournée	INT(4) TRUE% si tout est OK, FALSE% si la vérification d'un contrôle échoue.		

Voir aussi SETFOCUS, DLG_CHECKCONTROLS

Fonction DLG_CHECKCONTROLS (Librairie NSMisc)

Permet de faire une vérification basique (vérification que les contraintes dans les Entry Field sont respectées, les types ...) sans envoyer l'événement CHECK aux contrôles.

Syntaxe	DLG_CHECKCONTROL (<i>dlg</i>)		
Paramètre	dlg	POINTER	I nom du Custom Control
Valeur retournée	INT(4) TRUE% si tout est OK, FALSE% si la vérification d'un contrôle échoue.		

Le code interne d'un Custom Control gérant un EXECUTED (et n'étant pas coché NOCHECKING) utilise cette fonction avant de passer l'EXECUTED au code de l'utilisateur du contrôle (ou aussi avant de faire certains traitement automatiques qui ne doivent se faire que si le check réussit).

Fonction GETCONTROLKIND ([Librairie NSMisc](#))

Permet de connaître le type du contrôle passé en paramètre (Entry-field, Push-Button, ...).

Syntaxe	GETCONTROLKIND (<i>ctrl</i>)		
Paramètre	ctrl	CONTROL	I nom du contrôle
Valeur retournée	INT(2)		

1. Cette fonction retourne la même valeur que le qualificateur dynamique.KIND.
2. Si le contrôle dont on veut connaître le type est un custom-control, son événement GETVALUE ne sera pas appelé.

Voir aussi [GETCONTROLSUBKIND](#)

Fonction GETCONTROLSUBKIND ([Librairie NSMisc](#))

Elle permet de connaître le sous type d'une bitmap : icône, push-button ou check box.

Syntaxe	GETCONTROLSUBKIND (<i>ctrl</i>)		
Paramètre	ctrl	CONTROL	I nom du contrôle

Valeur retournée	INT(2) Cette valeur est une des constantes suivantes : <u>DI_ICON%</u> , <u>DI_PUSHBUTTON%</u> ou <u>DI_CHECKBOX%</u>
-------------------------	---

Cette fonction retourne la même valeur que le qualificateur dynamique .SUBKIND.
Voir aussi GETCONTROLKIND, DI *%

Fonction DEV_ADJUSTX (Librairie NSMisc)

Cette fonction transforme une valeur en son équivalent dans le nouveau système de coordonnées. Cette transformation prend en compte la résolution de la machine de développement et le facteur de zooming horizontal des fenêtres à l'exécution.

Syntaxe	DEV_ADJUSTX (win%, x%)		
Paramètres	win%	POINTER	I handle de la fenêtre pour laquelle on veut un ajustement.
	x%	INTEGER	I valeur dont on désire l'ajustement.
Valeur retournée	INTEGER valeur après ajustement		

Exemple :

```
PB_TEST.X = DEV_ADJUSTX (self%,50)
PB_TEST.WIDTH = DEV_ADJUSTX (self%,100)
;Dans cet exemple, les tailles et positions du push-button en X seront toujours correctes
quels que soient les différences de résolution et de tailles des fenêtres entre le
développement et l'exécution.
```

Voir aussi DEV_ADJUSTY

Fonction DEV_ADJUSTY (Librairie NSMisc)

Cette fonction transforme une valeur en son équivalent dans le nouveau système de coordonnées. Cette transformation prend en compte la résolution de la machine de développement et le facteur de zooming vertical des fenêtres à l'exécution.

Syntaxe	DEV_ADJUSTY (win%, y%)		
Paramètres	win%	POINTER	I handle de la fenêtre pour laquelle on veut un ajustement.
	y%	INTEGER	I valeur dont on désire l'ajustement.

Valeur retournée	INTEGER valeur après ajustement
-----------------------------	------------------------------------

Exemple :

```
PB_TEST.Y = DEV_ADJUSTY (self%,40)
PB_TEST.HEIGHT = DEV_ADJUSTX (self%,24)
;Dans cet exemple, les tailles et positions du push-button en Y seront toujours correctes
quels que soient les différences de résolution et de tailles des fenêtres entre le
développement et l'exécution.
```

Voir aussi DEV_ADJUSTX

Instruction SETZORDER (Librairie NSMisc)

Permet de modifier la position d'une fenêtre dans l'axe des "Z" (c'est à dire dessus, dessous une autre, ou au dessus de toutes les autres).

Syntaxe	SETZORDER win%, kind%, behind%		
Paramètres	win%	POINTER	handle de la fenêtre
	kind%	INTEGER	position de la fenêtre
	behind%	POINTER	handle de la deuxième fenêtre si le paramètre kind% vaut <u>ZOK_BEHIND%</u>

1. Déterminer la position de la fenêtre dans le paramètre kind% avec les constantes ZOK_*.
2. Si kind% ne contient pas la valeur ZOK_BEHIND%, le paramètre behind% est ignoré. Si kind% contient la valeur ZOK_BEHIND%, la fenêtre est positionnée par rapport à la fenêtre désignée dans le paramètre behind%.
3. Le paramètre ZOK_TOPMOST% est reconnu comme ZOK_ TOP% et le paramètre ZOK_NOTOPMOST% n'a aucun effet.

Voir aussi Constantes ZOK_*

Constantes ZOK_* (Librairie NSMisc)

Valeurs représentant la position de la fenêtre dans l'ordre Z et correspondant au paramètre kind% de l'instruction SETZORDER.

Syntaxe	Déclaration interne	Description
ZOK_TOP%	0	Indique que la fenêtre doit passer en avant-plan.
ZOK_TOPMOST%	1	Indique que la fenêtre doit passer en avant-plan et y rester même si elle n'a plus le focus.
ZOK_NOTOPMOST%	2	Désactive le mode ZOK_TOPMOST%.
ZOK_BOTTOM%	3	Indique que la fenêtre doit passer en arrière-plan.
ZOK_BEHIND%	4	Indique que la fenêtre doit passer derrière une autre fenêtre.

Si le paramètre kind% de l'instruction SETZORDER ne contient pas la valeur ZOK_BEHIND%, le paramètre behind% est ignoré. Si kind% contient la valeur ZOK_BEHIND%, la fenêtre est positionnée par rapport à la fenêtre désignée dans le paramètre behind% de l'instruction SETZORDER.

Voir aussi SETZORDER

Fonction GETDESKTOPHEIGHT% (Librairie NSMisc)

Permet de récupérer la largeur du bureau Windows en excluant la ou les barres d'outils (Démarrer ou autre) éventuellement collée sur les côtés haut et/ou bas.

Syntaxe	GETDESKTOPHEIGHT%
Valeur retournée	INT(4)

Voir aussi GETDESKTOPWIDTH%, GETDESKTOPX%, GETDESKTOPY%

Fonction GETDESKTOPWIDTH% (Librairie NSMisc)

Permet de récupérer la largeur du bureau Windows en excluant la ou les barres d'outils (Démarrer ou autres) éventuellement collée sur les côtés droit et/ou gauche.

Syntaxe	GETDESKTOPWIDTH%
Valeur retournée	INT(4)

Voir aussi [GETDESKTOPHEIGHT%](#), [GETDESKTOPX%](#), [GETDESKTOPY%](#)

Fonction GETDESKTOPX% ([Librairie NSMisc](#))

Permet de récupérer le premier pixel à gauche de l'écran qui fait partie du bureau (après d'éventuelles barres d'outils à gauche).

Syntaxe	GETDESKTOPX%
Valeur retournée	INT(4)

Voir aussi [GETDESKTOPWIDTH%](#), [GETDESKTOPHEIGHT%](#), [GETDESKTOPY%](#)

Fonction GETDESKTOPY% ([Librairie NSMisc](#))

Permet de récupérer le premier pixel en bas de l'écran qui fait partie du bureau (après d'éventuelles barres d'outils en bas).

Syntaxe	GETDESKTOPY%
Valeur retournée	INT(4)

Pour rappel, le 0 est en bas en NCL.

Voir aussi [GETDESKTOPWIDTH%](#), [GETDESKTOPHEIGHT%](#), [GETDESKTOPX%](#)

Fonction INSTALL_CALLBACK ([Librairie NSMisc](#))

Retourne un handle (hCallback) qui doit être passé à [CALL_PREV CALLBACK](#) et [REMOVE_CALLBACK](#).

Syntaxe	INSTALL_CALLBACK (pfnCallBack)		
Paramètre	pfnCallBack	POINTER	I pointeur sur une fonction
Valeur retournée	POINTER		

1. La fonction dont le pointeur est passé en paramètre doit être prototypée de la façon suivante. Le nom de la fonction peut être différent :

```
FUNCTION MyCallback(POINTER mySelf%, INTEGER Id%, INTEGER Msg%, POINTER Parm1, POINTER Parm2) RETURN INTEGER
```

La fonction MyCallback est alors appelée pour chaque événement concernant une fenêtre ou un contrôle de l'application, mySelf% et Id% indiquant la fenêtre ou le contrôle concerné, Msg% l'événement et Parm1 et Parm2 les paramètres associés.

2. Tout passage dans MyCallback doit provoquer un appel et un seul à CALL_PREV CALLBACK

```
i% = CALL_PREV CALLBACK(hCallback, mySelf%, Id%, Msg%, Parm1, Parm2)
;... traitement complémentaire
RETURN i%
ou
RETURN CALL_PREV CALLBACK(hCallback, mySelf%, Id%, Msg%, Parm1, Parm2)
```

Voir aussi [CALL_PREV CALLBACK](#)

Instruction REMOVE_CALLBACK (Librairie NSMisc)

Retire la callback ajoutée par [INSTALL_CALLBACK](#).

Syntaxe	REMOVE_CALLBACK hCallback		
Paramètre	hCallback	POINTER	I handle envoyé par INSTALL_CALLBACK

Voir aussi [INSTALL_CALLBACK](#) , [CALL_PREV CALLBACK](#)

Fonction CALL_PREV CALLBACK (Librairie NSMisc)

La fonction CALL_PREV CALLBACK passe le contrôle à une autre callback ajoutée par [INSTALL_CALLBACK](#) ou au traitement standard des événements.

Syntaxe	CALL_PREV CALLBACK (hCallback, win, id, msg, parm1, parm2)			
Paramètres	hCallback	POINTER	I	handle envoyé par INSTALL_CALLBACK
	win	POINTER	I	handle de la fenêtre
	id	INTEGER	I	identifiant de la fenêtre ou du contrôle
	msg	INTEGER	I	événement
	parm1	POINTER	I	paramètre associé à l'événement
	parm2	POINTER	I	paramètre associé à l'événement
Valeur retournée	INTEGER			

Voir aussi [INSTALL_CALLBACK](#) , [REMOVE_CALLBACK](#)

Fonction nsStrCmp% (Librairie NSMisc)

Compare et trie deux chaînes de caractères.

Syntaxe	nsStrCmp% (<i>S1\$, S2\$, IgnoreCase%</i>)			
Paramètres	S1\$	CSTRING		première chaîne de caractères
	S2\$	CSTRING		seconde chaîne de caractères
	IgnoreCase%	INT(2)		booléen permettant de faire la distinction (ou pas) entre majuscules et minuscules
Valeur retournée	INTEGER			

La fonction trie par ordre alphabétique et quand le paramètre IgnoreCase% est positionné à TRUE%, les majuscules sont prioritaires sur les minuscules.Exemple de liste triée avec la fonction nsStrCmp% :

Liste originale	Liste triée avec la fonction nsStrCmp%
-----------------	--

Papy	le Gendre
Pépé	le gendre
Père	Le Guenn
Pêche	le Guenn
Peccadille	Léa
Pierre	Legendre
Portrait	Leguenn
public	Papy
Legendre	Peccadille
le gendre	Pêche
le Gendre	Pépé
leguenn	Père
le Guenn	Pierre
Le Guenn	Portrait
Léa	public

Fonction **GetHeapArrayCount%** (Librairie NSMisc)

Retourne le nombre d'éléments alloués avec l'instruction NEW.

Syntaxe	GetHeapArrayCount% (PNewAllocatedVarArray)		
Paramètre	PNewAllocatedVarArray	POINTER	I index du paramètre
Valeur retournée	INTEGER 0 si aucun nombre d'éléments n'a été spécifié dans l'instruction NEW		

1. Il est impératif de déclarer au préalable un pointeur typé sur un tableau sans taille, puis d'allouer ce tableau avec la syntaxe suivante NEW [nbElements%]@Tableau%. Si ces opérations préalables n'ont pas été effectuées, l'instruction GetHeapArrayCount% peut planter ou retourner un résultat aléatoire.

Exemple de syntaxe de l'instruction NEW :

```
Local @MonTableau[], n%, UnTableauDeTailleFixe%[10]
... ; doit calculer n%
New [n%]@MonTableau%
... ; fait ce qu'il faut du tableau
TraiterTableau MonTableau% ; OK, pas besoin de conserver ou passer n%
TraiterTableau UnTableauDeTailleFixe%
; !!! peut planter ou retourner un résultat aléatoire !!!
...
Dispose @MonTableau% ; on en a terminé avec ce tableau
```

2. Le fonctionnement est identique avec les segments terminés par un champ qui est un tableau sans taille (et non un pointeur sur un tableau sans taille), sachant qu'il est interdit de faire des tableaux de ce genre de segment ou de les déclarer comme variable locale ou gloable, ou encore de les utiliser comme champs d'un autre segment sauf si c'est le dernier (et les mêmes restrictions s'appliquent alors à ce nouveau segment et ainsi de suite), sauf en tant que pointeurs typés sur le segment (qui peuvent être utilisés n'importe où), il faut impérativement allouer ces pointeurs typés par NEW.

Exemple de syntaxe de segment :

```
Segment Test
  Id%
  Valeurs%[] ; ce n'est pas un pointeur et c'est un tableau sans taille => ; doit être
le tout dernier champ du segment
EndSegment ; Test
...
Local Test@ t
...
New [10]@t ; segment Test ayant un tableau Valeurs% de 10 éléments
; GetHeapArrayCount%(@t) retournerait 10
...
```

Exemple :

```
for i% = 0 to GetHeapArrayCount%(@lib_hbook0.value.r) - 1
  auteur$ = lib_hbook0.value.r[i%].value.author
  ;things_trace "auteur : " & auteur$
  message "auteur", auteur$
  isbn$ = lib_hbook0.value.r[i%].value.isbn
  ;things_trace "auteur : " & auteur$
  message "isbn", isbn$
  title$ = lib_hbook0.value.r[i%].value.title
  ;things_trace "auteur : " & auteur$
  message "titre", title$
endfor
```

Voir aussi **NEW (Langage NCL)**

Instruction **GETCONTROLTEMPLATE** (Librairie NSMisc)

Cette instruction permet d'envoyer des événements et des paramètres dynamiques (via le paramètre *tpl*) au modèle générique de l'instance du Custom Control indiqué dans le paramètre *ctrl*.

Syntaxe	GETCONTROLTEMPLATE <i>ctrl, tpl</i>			
Paramètres	<i>ctrl</i>	CONTROL	I	nom de l'instance du Custom Control
	<i>tpl</i>	CONTROL	O	variable

Exemple :

```
; PPB1 est un contrôle de type Picture Button
LOCAL CONTROL CT

GetControlTemplate PPB1, CT
Message ControlName$(CT), ControlName$(PPB1)
;affiche une boîte de message avec "PICTBUT" (le nom générique du Custom Control) dans le
titre et "PPB1" dans la boîte (le nom du contrôle). PPB1 pourrait tout aussi bien être
remplacé par une variable de type CONTROL (à condition de l'avoir initialisée)
```

Codes d'erreur NSMisc

Cette annexe fournit les différents codes d'erreur pouvant être retournés par les fonctions MISCERROR%, T_ERROR% et F_ERROR% de la librairie NSMisc. Ces codes d'erreur sont donnés sous la forme de leur déclaration interne.

Liste des codes d'erreur NSMisc

Syntaxe	Déclaration interne
NO_ERROR%	0
ERROR_INVALID_FUNCTION%	1
ERROR_INVALID_HANDLE%	2
ERROR_NOT_ENOUGH_MEMORY%	3
ERROR_CANNOT_ALLOC_MEMORY%	4
ERROR_CANNOT_FREE_MEMORY%	5
ERROR_OUT_OF_PAPER%	6
ERROR_INVALID_PARAMETER%	7

ERROR_INVALID_DATA%	8
ERROR_INVALID_NAME%	9
ERROR_DISK_FULL%	10
ERROR_WRITE_PROTECT%	11
ERROR_HARDWARE_FAULT%	12
ERROR_SHARING_VIOLATION%	13
ERROR_LOCK_VIOLATION%	14
ERROR_INVALID_FILE_SPECIFICATION%	15
ERROR_TOO_MANY_OPENED_FILES%	16
ERROR_ACCESS_DENIED%	17
ERROR_FILE_EXISTS%	18
ERROR_FILE_NOT_FOUND%	19
ERROR_FILE_NOT_OPENED%	20
ERROR_BEYOND_END_OF_FILE%	21
ERROR_CANNOT_CREATE_FILE%	22
ERROR_CANNOT_ACCESS_TO_FILE%	23
ERROR_CANNOT_OPEN_FILE%	24
ERROR_CANNOT_READ_FILE%	25
ERROR_CANNOT_WRITE_ON_FILE%	26
ERROR_CANNOT_CLOSE_FILE%	27
ERROR_NO_FILE_FOUND%	28
ERROR_CANNOT_UNLOCK_FILE%	29
ERROR_LOCK_FAILED%	30
ERROR_NOT_LOCKED%	31
ERROR_BAD_FORMAT%	32
ERROR_QUEUE_EMPTY%	33
ERROR_CANNOT_EXEC_PROG%	34
ERROR_CANNOT_CHANGE_DIR%	35
ERROR_CANNOT_MAKE_DIR%	36

ERROR_CANNOT_REM_DIR%	37
ERROR_CANNOT_GET_DIR%	38
ERROR_CANNOT_CHANGE_DISK%	39
ERROR_CANNOT_GET_DISK%	40
ERROR_CANNOT_REN_FILE%	41
ERROR_CANNOT_REM_FILE%	42
ERROR_NO_MORE_CONTROL%	43
ERROR_CANNOT_ACCESS_TO_QUEUE%	44
ERROR_CANNOT_CREATE_QUEUE%	45
ERROR_INVALID_UNIT%	46
ERROR_CANNOT_LOAD_MODULE%	47
ERROR_CANNOT_GET_PROC_ADDRESS%	48
ERROR_FAILED_TO_COPY_FILE%	49

Formats d'affichage

Cette annexe décrit les formats pouvant être employés dans les fonctions STRING\$ et ESTRING\$ de la librairie NSMisc.

Formats de dates

Les formats date s'appuient sur une valeur numérique correspondant au nombre de jours écoulés depuis le 1er janvier 1900. Pour les dates antérieures au 1er janvier 1900, la valeur numérique est négative, et il est possible de remonter jusqu'à la naissance du calendrier Grégorien c'est-à-dire jusqu'au 1er janvier 1583.
Exemple :

```
1.01.1900 0
2.01.1900 1
23.01.1992 33624
31.12.1899 -1
1.01.1583 -115782
```

Cette méthode de conversion permet tous les calculs, évaluations, durées... En effet toutes les dates affichées peuvent être converties en nombre de jours, évalué

d'après le 1er janvier 1900, être soumises à un calcul et être reconverties en format date.

Vous pouvez saisir la date de quelque manière que ce soit à partir du moment où elle suit une des règles de format proposées, la conversion s'effectuera selon le format spécifié.

Par exemple, avec le format d/m/yy, si vous saisissez 12041965, vous obtiendrez 12/4/65 à l'affichage.

Formats d'heures

De la même manière, les formats heures s'appuient sur une valeur numérique correspondant au nombre de secondes écoulées depuis 0 heure (minuit).

Exemple:

```
0:00:00 0
12:00:00 43200
23:59:59 86399
```

Le tableau de la page suivante indique les caractères pouvant être spécifiés dans la description du format et leur signification.

Les dernières pages de l'annexe vous donnent quelques exemples de formats usuels et l'affichage qui en découle.

Formats et spécifications

Format	Signification
0	<p>Signifie un chiffre.</p> <p>Si le nombre saisi contient moins de chiffres que le format n'en prévoit, des zéros seront affichés.</p> <p>Si le nombre saisi comporte plus de chiffres après le séparateur décimal que le format n'en prévoit, le nombre sera tronqué pour s'adapter au format spécifié, c'est-à-dire arrondi au chiffre immédiatement supérieur ou inférieur.</p> <p>Si le nombre que vous saisissez comporte plus de chiffres avant le séparateur décimal que le format n'en prévoit, le nombre sera affiché quand même dans son intégralité</p>

#	<p>Signifie un chiffre et supprime les éventuels zéros inutiles.</p> <p>Si des zéros figurent devant les chiffres placés avant le séparateur décimal, ceux-ci seront éliminés.</p> <p>De même si des zéros figurent après les chiffres composant la décimale, ils seront éliminés.</p>
.	<p>Caractère marquant la décimale.</p> <p>Prévoir au moins un format 0 après le séparateur pour éviter l'affichage d'un point après le nombre.</p> <p>Prévoir au moins un format 0 avant le séparateur pour éviter l'affichage des chiffres inférieurs à 1 précédé d'un seul point.</p> <p>Si le format prévu après le séparateur est plus petit que le nombre saisi, celui-ci sera arrondi.</p> <p>Si le format prévu avant le séparateur est plus petit que le nombre saisi, celui-ci sera affiché dans son intégralité.</p>
%	<p>Symbole et opérateur du pourcentage.</p> <p>Multiplie le nombre saisi par 100 et ajoute le caractère %.</p>
espace	<p>Permet la séparation des milliers si le format possède un caractère # ou 0 avant et après l'espace.</p> <p>Si le format ne contient qu'un espace représentant le premier millier, et que nombre est supérieur à 4 chiffres, un seul espace sera inséré conformément au format.</p>

;	<p>Lorsque le format doit être différent selon que le chiffre est positif, négatif ou égal à 0, séparer les deux ou trois formats par un ";".</p> <p>Le premier format considèrera les nombres positifs, le second les nombres négatifs et le troisième 0.</p> <p>Si un format est spécifié et suivi d'un point virgule, les nombres négatifs seront ignorés.</p> <p>Si un format est spécifié et suivi de deux points-virgule, les nombres négatifs et égaux à zéros seront ignorés.</p> <p>Un ";" positionné après les formats des chiffres positifs et négatifs, les valeurs nulles seront ignorées.</p> <p>Deux ";" en tant que format aura pour effet d'ignorer n'importe quel nombre.</p>
[Red]	Met en rouge le texte qui le suit.
E-E+e-e+	<p>Format de notation scientifique.</p> <p>Si, à droite du séparateur décimale figure un 0 ou un #, un E ou un e sera affiché afin d'exprimer l'exposant.</p> <p>Le nombre de caractères de format spécifiés à droite du séparateur détermine le nombre de chiffres en exposant.</p> <p>Spécifier un signe + ou - juste après le E ou le e pour visualiser ce signe dans le nombre affiché.</p>
:eE-+() espace	Pour afficher un caractère autre que ceux-ci, il faut utiliser des guillemets suivis du caractère voulu.
"texte"	La saisie de n'importe quel caractère provoquera l'affichage des caractères spécifiés entre les guillemets.
M	Affiche le mois sous forme de nombre sans zéro précédant les chiffres de 1 à 9.
MM	Affiche le mois sous forme de nombre avec un zéro précédant les chiffres de 1 à 9.

mmm ou MMM	Affiche le mois sous la forme abrégée (jan, feb, mar...).
mmmm ou MMMM	Affiche le mois sous sa forme complète (January, February...).
d	Affiche le jour de la semaine sous forme de chiffre non précédé d'un zéro.
dd	Affiche le jour de la semaine sous forme de chiffre précédé d'un zéro
ddd	Affiche le jour de la semaine sous la forme abrégée (Mon, Tue...).
dddd	Affiche le jour de la semaine sous sa forme complète (Monday, Tuesday...).
yy	Affiche l'année sous forme d'un nombre à deux chiffres de 00 à 99.
yyyy	<p>Affiche l'année sous sa forme complète de 4 chiffres (1900...2040). / Caractère reconnu par le format de date ou d'heure comme séparateur.</p> <p>Si la date est saisie sous un format date, celle-ci sera affichée conformément.</p> <p>Si la date est saisie sous forme de nombre, celui-ci sera considéré comme le nombre de jour depuis le 1^{er} janvier 1900 et converti.</p>
h	Affiche l'heure sans afficher de zéros précédant les chiffres de 0 à 9.
hh	Affiche l'heure précédée d'un zéro avant les chiffres de 0 à 9.
m	Affiche les minutes sans afficher de zéros précédant les chiffres de 0 à 9.
mm	Affiche les minutes précédées d'un zéro avant les chiffres de 0 à 9.
s	Affiche les secondes sans afficher de zéros précédant les chiffres de 0 à 9.
ss	Affiche les secondes précédées d'un zéro avant les chiffres de 0 à 9.

A propos des formats pour STRING\$ et ESTRING\$

Vous pouvez choisir un format dans la liste pré-établie, modifier l'un d'eux ou en créer un avec les caractères et symboles ci-dessous.

La vérification sur les caractères saisis et l'attribution du format s'effectuent lorsque le champ éditable perd le focus.

Exemples

Format	Saisie	Résultat
0	23	23
00.000	2.5	02.500
0	-3	-3
0.##	34	34.
	5.1236	5.12
	1.168	1.17
0.0	4.58	4.6
##.##	2.5	2.5
##.##	0.1	.1
##.##	34.1254	34.13
#,000.##	1,321.147	1,321.15
\$#,##0;(\$#,##0)	3,125	\$3,125 (\$3,125)
	-3,125	
# ###.00	4567	4 567.00
# ##0 \$	3254	3 254 \$
;;	4587.12	aucun nombre affiché
00-00-00	123456	12-34-56
0%	0.1234	12%
0.00%	0.1234	12.34%
0.00E+00	1234.568	1.23E+03
\$##.##	12.5	\$12.5
d-MM-yy	31291	1-09-85

d-mmm-yy	2/12/91	2-dec-91
dd-MM-yyyy	2-12-91	02/12/1991
d mmm yyyy	12/07/91	12 jul 1991
hh:mm	13:25	1:25

LIBRAIRIE NSDYNSTR

Installation

Déclarez NSDYNSTR, NSERRORS et NSMISC dans les librairies nécessaires au développement de votre application.

Vérifiez que les fichiers NSxxDYNSTR.DLL, NSxxERRORS.DLL et NSxxMISC.DLL sont bien dans un des répertoires PATH sous Windows.

Fichier NSDynstr.NCL

Les verbes de la librairie NSDynstr, décrits ci-après, sont déclarés dans le fichier texte écrit en NCL, de nom NSDYNSTR.NCL. Ce fichier peut aussi contenir des verbes complémentaires (API non publique). Vous pouvez donc désirer consulter ce fichier pour avoir la référence exhaustive de la librairie.

Pour consulter le fichier NSDYNSTR.NCL:

1. Placez-vous dans le répertoire <NATSTAR>\NCL.

<NATSTAR> représente le répertoire que vous avez choisi au moment de l'installation de NatStar. La procédure est identique pour NS-DK et NatWeb.

2. Editez le fichier NSDYNSTR.NCL avec n'importe quel éditeur de texte.

Référence de la librairie NSDynstr

Fonction IsDSNull% (Librairie NSDynstr)

Permet de savoir si une chaîne de caractères a été modifiée ou non.

Syntaxe	IsDSNull% (ds)		
Paramètre	ds	DYNSTR	I chaîne de caractères
Valeur retournée	TRUE% ou FALSE%		

1. La valeur retournée est un booléen indiquant s'il s'agit d'une variable modifiée ou non depuis sa création ou depuis le dernier appel à SetDSNull sur cette variable.

```
Local DynStr ds
```

```
Message "ds est nulle", IsDSNull%(ds)  
; retourne TRUE%, affiché comme 1  
ds = "" ; chaîne vide mais néanmoins modifiée
```

```
Message "ds n'est plus nulle", IsDSNull%(ds)
; retourne FALSE%, affiché comme 0
```

2. Si on écrit `ds=F$(paramètres)` et que `F$` retourne une DynStr nulle (mais pas une chaîne vide, et `F$` doit être déclarée comme retournant une DynStr), alors `IsDSNull%(ds)` retourne `TRUE%`, même si la DynStr retournée est l'un des paramètres (DynStr) de `F$` qui aurait reçu une variable DynStr nulle. Cela n'est pas vrai pour `Copy$`, `Delete$`, `Insert$`, `Skip`, `LSkip`, `RSkip`, `UpCase`, `LowCase`, `&` et `&&`, même si on leur passe une (ou deux pour `Insert$`, `&` et `&&`) DynStr nulle(s), ces fonctions retournent toujours une DynStr non nulle.

3. Le texte d'une DynStr nulle est identique à celui d'une DynStr vide, même pour les opérateurs de comparaison, seule `IsDSNull%` peut les distinguer. Cela a été fait pour pouvoir diagnostiquer une variable DynStr non initialisée.

Voir aussi [SetDSNull](#)

Instruction SetDSNull (Librairie NSDynstr)

Permet de positionner une variable DynStr à l'état NULL et supprime également son contenu.

Syntaxe	SetDSNull ds			
Paramètre	ds	DYNSTR	O	chaîne de caractères

Voir aussi [IsDSNull%](#)

Fonction ForceDSLength (Librairie NSDynstr)

Permet de forcer la taille d'une chaîne de caractères.

Syntaxe	ForceDSLength (ds, Len%)			
Paramètres	ds	DYNSTR	I	chaîne de caractères
	Len%	INTEGER	I	longueur de la chaîne de caractères
Valeur retournée	POINTER			

1. La fonction `ForceDSLength` modifie la chaîne de caractères `ds` pour qu'elle puisse contenir le nombre de caractères requis (plus un caractère pour le zéro de fin de chaîne, `Len%` indique le nombre de caractères utiles). `Len%` qui contient la longueur demandée en entrée, prend la longueur disponible en sortie. IL EST IMPÉRATIF de ne pas utiliser plus de caractères que la valeur

contenue dans Len% en sortie (plus l'octet du zéro de fin de chaîne qui est déjà présent à la bonne position après l'appel à ForceDSLenght).

2. La fonction retourne un pointeur sur le premier caractère (non initialisé) de la chaîne. Il faut utiliser ce pointeur pour remplir la chaîne de caractères. Les caractères initiaux de la chaîne peuvent être perdus.

3. La longueur Len% peut être raccourcie en sortie dans deux cas : si la taille demandée est trop grande, ou bien si la variable DynStr passée est en fait un paramètre de fonction de type DynStr@ auquel on aurait passé une variable de type CSTRING(n) au lieu d'une DynStr (ce qui est permis mais limite alors la taille de la variable) lors d'un appel à cette fonction, auquel cas la taille de cette variable ne peut dépasser n (255 si non spécifié).

4. Cette fonction n'est à utiliser que si l'on souhaite transférer le plus efficacement possible une suite de buffers de caractères dont on connaît la taille totale à l'avance mais que l'on ne peut obtenir en une seule fois (pour éviter des concaténations qui réallouent et copient la DynStr à chaque fois), ou alors si on a besoin d'un pointeur direct sur le buffer mémoire des caractères de la DynStr pour éviter une copie et/ou des concaténations (par exemple, si on veut faire un ou plusieurs F_BLOCKREAD pour transférer un fichier texte (sans 0 de fin de chaîne au milieu) dans une DynStr.

Exemple :

```
; Lit le contenu du fichier texte dont le nom est FileName$ dans FileContent$.
; Retourne l'un des codes d'erreur de NSERRORS.NCL, No_Error% si OK.
Function LoadFile%(DynStr FileName$, DynStr@ FileContent$)
  Local F%, Ret%(1), Cnt%(2), Integer SubSize%, Integer Size%, Pointer P

  F% = F_R00pen%(1, FileName$)
  Ret% = F_Error%
  If Ret% = No_Error%
    Size% = F_Size%(F%)
    Ret% = F_Error%
    If Ret% = No_Error%
      ; essaye de forcer la longueur de la DynStr FileContent$ à la taille du fichier
      FileName$, un octet de plus est alloué et initialisé à 0 pour la fin de la chaîne. Size%
      peut être modifié si la DynStr FileContent$ pointe en fait une CString(n%) et ne peut
      prendre la taille souhaitée. Obtient un pointeur sur la zone allouée qu'il faut alors
      remplir avec du texte.
      P = ForceDSLenght(FileContent$, Size%)
      If P = 0
        Ret% = Error_Not_Enough_Memory% ; échec de ForceDSLenght
      Else
        SubSize% = $4000 ; lit 16ko à la fois
      ; tant qu'il y a de la place et qu'on atteint pas la fin du fichier
      While (Size% > 0) And Not F_EOF%(F%)
        If Size% < SubSize%
          SubSize% = Size%
      ; dernier bloc lu plus petit que 16ko
      EndIf
      F_BlockRead F%, P, SubSize%, Cnt%
      ; lecture du bloc, Cnt% octets lus
      Ret% = F_Error%
```

```

    If Ret% <> No_Error%
        Break ; erreur de lecture du fichier
    EndIf
    P = P + Cnt%
; avance le pointeur sur le bloc suivant
    Size% = Size% - Cnt%
; déduit la taille lue de la taille allouée
    EndWhile
    If Size% > 0
; on a lu moins de caractères qu'on en a alloué ?
        ; Tronque la DynStr au nombre de caractères lus
        FileContent$ = Copy$(FileContent$, 1, Length(FileContent$) - Size%)
    EndIf
    EndIf
    EndIf
    F_Close F%
    EndIf
    Return Ret%
EndFunction ; LoadFile$

```

Voir aussi IsDSNull%, SetDSNull

Exemple de sauvegarde du contenu d'une chaîne dans un fichier (Librairie NSDynstr)

```

; Sauve le contenu de la chaîne FileContent$ dans le fichier dont le nom est dans
; FileName$. Retourne l'un des codes d'erreur de NSERRORS.NCL, No_Error% si OK.
Function SaveFile%(DynStr FileName$, DynStr FileContent$)
    Local F%, Ret%(1), Cnt%(2), SubSize%, Size%, Pointer P

    F% = F_Create%(1, FileName$)
    Ret% = F_Error%
    If Ret% = No_Error%
        Size% = Length(FileContent$) ; taille du fichier
        ; obtient un pointeur sur les caractères existants de la DynStr, ON NE DOIT JAMAIS LES
        MODIFIER CAR PLUSIEURS DYNSTR PEUVENT PARTAGER LE MÊME TEXTE EN MÉMOIRE, sauf
        immédiatement après un ForceDSLenght$, tant que la DynStr qu'on lui a passé n'est pas
        copiée ou passée en paramètre à une fonction. D'autre part, CE POINTEUR PEUT DEVENIR
        INVALIDE si on change la DynStr.
        P = @FileContent$
        SubSize% = $4000 ; écrit 16ko à la fois
        While Size% > 0
            If Size% < SubSize%
                SubSize% = Size%
            ; dernier bloc écrit plus petit que 16ko
            EndIf
            F_BlockWrite F%, P, SubSize%, Cnt%
; écriture du bloc, Cnt% octets écrits
            Ret% = F_Error%
            If Ret% <> No_Error%
                Break
            EndIf
            If SubSize% <> Cnt%
; nombre d'octets écrits plus petit que celui demandé ?
                Ret% = Error_Cannot_Write_On_File%
                Break
            EndIf
            P = P + SubSize% ; avance le pointeur sur le bloc suivant
            Size% = Size% - SubSize% ; déduit la taille écrite de la ;longueur totale
        EndWhile
        F_Close F%
    EndIf
EndFunction

```



```
Return Ret%
EndFunction ; SaveFile%

; Comme T_ReadLn$ mais sans limitation à 255 caractères !
Function T_ReadLnDS$(TF%) Return DynStr
Local DynStr Ret$

Ret$ = T_Read$(TF%)
While (T_Error% = No_Error%) And Not T_EOL%(TF%)
    Ret$ = Ret$ & T_Read$(TF%)
EndWhile
Ret$ = Ret$ & T_ReadLn$(TF%)
Return Ret$
EndFunction ; T_ReadLnDS$
```


LIBRAIRIE NSMLE

Cette librairie contient des fonctions permettant de manipuler le texte dans les contrôles ou les fenêtres de type MLE.

Installation (Librairie NSMLE)

Déclarez NSMLE dans les librairies nécessaires au développement de votre application.

Vérifiez que le fichier NSxxMLE.DLL est bien dans un des répertoires PATH sous Windows.

Fichier NSMLE.NCL (Librairie NSMLE)

Les verbes de la librairie NSMLE, décrits ci-après, sont déclarés dans le fichier texte écrit en NCL, de nom NSMLE.NCL. Ce fichier peut aussi contenir des verbes complémentaires (API non publique). Vous pouvez donc désirer consulter ce fichier pour avoir la référence exhaustive de la librairie.

Pour consulter le fichier NSMLE.NCL :

1. Placez-vous dans le répertoire <NATSTAR>\NCL.

<NATSTAR> représente le répertoire que vous avez choisi au moment de l'installation de NatStar. La procédure est identique pour NS-DK et NatWeb.

2. Editez le fichier NSMLE.NCL avec n'importe quel éditeur de texte.

Vous pouvez également consulter le fichier NSMLE.NCL à partir de NS-Design.

Référence de la librairie NSMLE

Instruction NSMLE_CUT (Librairie NSMLE)

Cette instruction permet de déplacer dans le presse-papiers la chaîne de caractères sélectionnée.

Syntaxe	NSMLE_CUT <i>ctrl</i>		
Paramètre	ctrl	CONTROL	I contrôle de type MLE

Exemple :

```
; ctrl est un contrôle de type mle
NSMLE_CUT ctrl
```

Voir aussi NSMLE_PASTE, NSMLE_COPY

Instruction **NSMLE_COPY** (Librairie **NSMLE**)

Permet de copier dans le presse-papiers la chaîne de caractères sélectionnée.

Syntaxe	NSMLE_COPY <i>ctrl</i>			
Paramètre	ctrl	CONTROL	I	contrôle de type MLE

Exemple :

```
; ctrl est un contrôle de type mle
NSMLE_COPY ctrl
```

Voir aussi [*NSMLE_PASTE*](#), [*NSMLE_CUT*](#)

Instruction **NSMLE_PASTE** (Librairie **NSMLE**)

Permet de recopier la chaîne se trouvant dans le presse-papiers.

Syntaxe	NSMLE_PASTE <i>ctrl</i>			
Paramètre	ctrl	CONTROL	I	contrôle de type MLE

Exemple :

```
; ctrl est un control de type mle
NSMLE_PASTE ctrl
```

Voir aussi [*NSMLE_CUT*](#), [*NSMLE_COPY*](#)

Instruction **NSMLE_CLEAR** (Librairie **NSMLE**)

Permet d'effacer la chaîne de caractères sélectionnée.

Syntaxe	NSMLE_CLEAR <i>ctrl</i>			
Paramètre	ctrl	CONTROL	I	contrôle de type MLE

Exemple :

```
; ctrl est un control de type mle
NSMLE_CLEAR ctrl
```

Voir aussi [*NSMLE_PASTE*](#), [*NSMLE_CUT*](#), [*NSMLE_COPY*](#), [*NSMLE_CLEAR*](#)

Fonction **NSMLE_LOAD%** (Librairie **NSMLE**)

Permet de charger le contenu d'un fichier.

Syntaxe	NSMLE_LOAD% (<i>ctrl</i> , <i>name\$</i>)
----------------	--

Paramètres	ctrl	CONTROL	I	contrôle de type MLE
	name\$	CSTRING	I	nom du fichier
Valeur retournée	INT(1)			

Exemple :

```
LOCAL CString filename
LOCAL int ret
ret = tas_dbox_getfilename (self%,d_tas_dbox_getfilename_open%,"Open file for
MLE","*.txt","Text Files" & "*.txt", filename)
NSMLE_LOAD%(mle1, filename)
```

Fonction NSMLE_SETTEXT% (Librairie NSMLE)

Permet d'écrire une chaîne de caractères à la position du curseur. Pour éviter la limitation à 256 caractères due au CString il faut passer par une variable dynstr qui elle ne sera pas limitée.

Syntaxe	NSMLE_SETTEXT% (<i>ctrl, string</i>)			
Paramètres	ctrl	CONTROL	I	contrôle de type MLE
	string	CSTRING	I	chaîne de caractères
Valeur retournée	INT(1)			

Le paramètre string peut être de type CSTRING ou DYNSTR (sans troncature).

Exemple :

```
LOCAL cstring chaine
Chaine = « test »
Message "NSMLE_SetText%", NSMLE_SetText%(ctrl,chaine, length(chaine))
```

Voir aussi [NSMLE_GETTEXT](#), [NSMLE_GETTEXTSIZE%](#)

Instruction NSMLE_GETTEXT (Librairie NSMLE)

Permet de récupérer le texte d'un contrôle MLE sous la forme d'une chaîne dynamique Dynstr.

Syntaxe	NSMLE_GETTEXT <i>ctrl</i>			
Paramètre	ctrl	CONTROL	I	contrôle de type MLE

Valeur retournée	DYNSTR
-------------------------	--------

Exemple :

```
LOCAL DYNSTR ds
Ds = NSMLE_GetText (ctrl)
```

Voir aussi [NSMLE_SETTEXT](#), [NSMLE_GETTEXTSIZE%](#)

Fonction [NSMLE_GETTEXTSIZE%](#) ([Librairie NSMLE](#))

Permet de récupérer la taille du texte contenu dans la fenêtre.

Syntaxe	NSMLE_GETTEXTSIZE% (ctrl)			
Paramètre	ctrl	CONTROL		contrôle de type MLE
Valeur retournée	INTEGER			

Exemple :

```
Local int sz
Sz = NSMLE_GetTextSize%(ctrl)
```

Voir aussi [NSMLE_GETTEXT](#)

Fonction [NSMLE_ISSELECTED%](#) ([Librairie NSMLE](#))

Permet de savoir s'il y a un bloc sélectionné.

Syntaxe	NSMLE_ISSELECTED% (ctrl)			
Paramètre	ctrl	CONTROL		contrôle de type MLE
Valeur retournée	INT(1)			

Exemple :

```
NSMLE_ISSELECTED% ctrl
```

Voir aussi [NSMLE_UNDO](#)

Fonction [NSMLE_ISUNDO%](#) ([Librairie NSMLE](#))

Permet de savoir s'il y a une action à annuler.

Syntaxe	NSMLE_ISUNDO% (ctrl)
----------------	-----------------------------

Paramètre	ctrl	CONTROL	I	contrôle de type MLE
Valeur retournée	INT(1)			

Exemple :

```
NSMLE_ISUNDO% ctrl
```

Voir aussi [NSMLE_UNDO](#)

Fonction NSMLE_ISCLIPBOARD% (Librairie NSMLE)

Permet de savoir s'il y a quelque chose dans le presse-papiers à recopier.

Syntaxe	NSMLE_ISCLIPBOARD% (ctrl)			
Paramètre	ctrl	CONTROL	I	contrôle de type MLE
Valeur retournée	INT(1)			

Exemple :

```
NSMLE_ISCLIPBOARD% ctrl
```

Voir aussi [NSMLE_CUT](#), [NSMLE_COPY](#)

Instruction NSMLE_UNDO (Librairie NSMLE)

Permet d'annuler la dernière action faite.

Syntaxe	NSMLE_UNDO ctrl			
Paramètre	ctrl	CONTROL	I	contrôle de type MLE

Exemple :

```
NSMLE_UNDO ctrl
```

Voir aussi [NSMLE_ISUNDO](#)

Fonction NSMLE_SELECTION\$ (Librairie NSMLE)

Permet de sélectionner une chaîne de caractères.

Syntaxe	NSMLE_SELECT\$ (ctrl)			
Paramètre	ctrl	CONTROL	I	contrôle de type MLE

Valeur retournée	CSTRING
-------------------------	---------

Exemple :

```
Pattern$ = NSMLE_SELECTION$(ctrl)
If
  NSMLE_SEARCH$(ctrl, Pattern$, WholeWord%, CaseSensitive%)
EndIf
```

Voir aussi [NSMLE_SEARCH%](#), [NSMLE_CHANGE%](#)

Fonction NSMLE_SEARCH% (Librairie NSMLE)

Permet d'effectuer une recherche dans le contenu d'un texte.

Syntaxe	NSMLE_SEARCH% (ctrl, pattern\$, wholeword%, casesensitive%)			
Paramètres	ctrl	CONTROL	I	contrôle de type MLE
	pattern\$	CSTRING	I	chaîne de caractères à rechercher
	wholeword%	INT(1)	I	booléen permettant d'indiquer si la recherche s'effectue sur l'intégralité de la chaîne de caractère
	casesensitive%	INT(1)	I	booléen permettant d'indiquer si la casse est sensible
Valeur retournée	INT(1)			

Exemple :

```
Pattern$ = NSMLE_SELECTION$(ctrl)
If NSMLE_SEARCH$(ctrl, Pattern$, WholeWord%, CaseSensitive%)
EndIf
```

Voir aussi [NSMLE_SELECTION\\$](#), [NSMLE_CHANGE%](#)

Fonction NSMLE_CHANGE% (Librairie NSMLE)

Permet de remplacer un texte par un autre.

Syntaxe	NSMLE_CHANGE% (ctrl, pattern\$, replaceby\$, wholeword%, casesensitive%, changedall%)
---------	---

Paramètres	ctrl	CONTROL	I	contrôle de type MLE
	pattern\$	CSTRING	I	chaîne de caractères à rechercher
	replaceby\$	CSTRING	I	chaîne de caractères de remplacement
	wholeword%	INT(1)	I	booléen permettant d'indiquer si la recherche s'effectue sur l'intégralité de la chaîne de caractère
	casesensitive%	INT(1)	I	booléen permettant d'indiquer si la casse est sensible
	changeall%	INT(1)	I	booléen permettant d'indiquer si le remplacement s'effectue sur toutes les occurrences rencontrées dans le texte
Valeur retournée	INT(1)			

Exemple :

```
NSMLE_CHANGE%(mylectrl, « search », « replaceby », wholeword%, casesensitive%, changeall%)
```

Voir aussi NSMLE_SELECTION\$, NSMLE_SEARCH%

LIBRAIRIE NSDYNCTRL

Cette librairie contient des fonctions permettant de manipuler dynamiquement les contrôles.

Installation et utilisation

Déclarez NSDYNCTRL dans les librairies nécessaires au développement de votre application.

Vérifiez que le fichier NSxxDYNC.DLL est bien dans un des répertoires PATH sous Windows.

Pour consulter le fichier NSDYNCTRL.NCL:

1. Placez-vous dans le répertoire <NATSTAR>\NCL ou <NSDK>\NCL
<NATSTAR> ou <NSDK> représentent le répertoire que vous avez choisi au moment de l'installation de NatStar ou NS-DK.

2. Editez le fichier NSDYNCTRL.NCL avec n'importe quel éditeur de texte.
Vous pouvez également consulter le fichier NSDYNCTRL.NCL à partir de NS-Design.

Référence de la librairie NSDYNCTRL

Constantes DCA_EF_* (Librairie NSDYNCTRL)

Ces constantes permettent d'implémenter des attributs de contrôles dynamiques (Dynamic Control Attributes) pour le contrôle Entry Field.

Syntaxe	Déclaration interne	Description
DCA_EF_AUTOSCROLL	1	Un défilement automatique du contenu du champ de saisie est effectué lorsque le nombre de caractères saisis dépasse la taille de ce champ. Les caractères devenus invisibles ne sont pas perdus. Le défilement s'effectue en cours de saisie.

DCA_EF_HORIZSCROLL	2	Permet d'afficher deux flèches horizontales à droite de l'Entry Field permettant de faire défiler le champ de saisie lorsque le nombre de caractères saisis dépasse la taille du champ.
DCA_EF_VERTSCROLL	4	Permet d'afficher deux flèches verticales dans l'Entry Field permettant de faire défiler le champ de saisie lorsque le nombre de caractères saisis dépasse la taille du champ.
DCA_EF_EXPLODING	8	Permet de modifier la hauteur de l'Entry Field en cours de saisie si le texte saisi dépasse la largeur du champ, afin que tous les caractères saisis soient visibles. En quittant la saisie, c'est-à-dire en quittant l'Entry Field, celui-ci reprend sa taille initiale.
DCA_EF_CENTER	16	Permet de justifier le texte au centre.
DCA_EF_LEFT	32	Permet de justifier le texte à gauche.
DCA_EF_RIGHT	64	Permet de justifier le texte à droite.
DCA_EF_MARGIN	128	Permet d'encadrer un champ de saisie.
DCA_EF_REQUIRED	256	Permet d'imposer une saisie dans l'Entry-Field.
DCA_EF_DISABLE	512	Permet de désactiver le contrôle.
DCA_EF_LOCKTEXT	1024	Permet de verrouiller la saisie d'un champ.
DCA_EF_FULLTEXT	2048	Permet d'obliger l'utilisateur à saisir le nombre de caractères spécifié dans le champ Max. length.
DCA_EF_UPCASE	4096	Permet de traduire automatiquement en majuscules les caractères saisis en minuscules.

DCA_EF_SKIPBLANKS	8192	Permet d'éliminer automatiquement les espaces saisis en début et en fin de champ.
DCA_EF_NOBLANKS	16384	Permet d'interdire la saisie d'espaces.
DCA_EF_OVER	32768	Permet d'afficher le contrôle en aspect bombé.
DCA_EF_BELOW	65536	Permet d'afficher le contrôle en aspect creux.
DCA_EF_AUTOTAB	131072	Pour passer automatiquement le focus au champ suivant.
DCA_EF_HIDETEXT	262144	Permet de saisir des mots de passe : les caractères saisis ne seront pas affichés, mais seront remplacés par des '*'. Cela permet de saisir des mots de passe complexes sans que l'utilisateur ne soit obligé de les taper à l'aveugle.
DCA_EF_INTEGER	524288	Impose la saisie d'un nombre entier.
DCA_EF_NUMBER	1048576	Impose la saisie d'un nombre réel.
DCA_EF_DATE	2097152	Impose la saisie d'une date.
DCA_EF_TIME	4194304	Impose la saisie d'une heure.
DCA_EF_HIDDEN	8388608	Permet de cacher le contrôle

Constantes DCA LB * (Librairie NSDYNCTRL)

Ces constantes permettent d'implémenter des attributs de contrôles dynamiques (Dynamic Control Attributes) pour le contrôle List Box.

Syntaxe	Déclaration interne	Description
DCA_LB_DISABLE	1	Permet de désactiver le contrôle.
DCA_LB_NOADJUSTPOS	2	Permet de ne pas ajuster la hauteur de la liste.
DCA_LB_MULTIPLESEL	4	Permet la sélection de plusieurs lignes.
DCA_LB_USERDRAW	8	

DCA_LB_HORIZSCROLLBAR	16	Permet d'afficher une barre de défilement horizontale dans une liste
DCA_LB_REALTIME	32	Permet que le déroulement de la liste soit synchrone avec le déplacement de l'ascenseur
DCA_LB_OVER	64	Permet d'afficher le contrôle en aspect bombé.
DCA_LB_BELOW	128	Permet d'afficher le contrôle en aspect creux.
DCA_LB_HIDDEN	8388608	Permet de cacher le contrôle

Exemple :

```
Local DC_ControlProperties@ Properties
LOCAL i%, j%, h%, h1%, h2%
SEND EXECUTED TO PB_DESTROY
i% = 2000
new @Properties
Properties.Kind=DI_LISTBOX%
Properties.DialogHandle=self%
Properties.ID =Firstid% + GetTotalDynamicControls%
Properties.X=4
Properties.Y=getClientheight% - 230 - GetTotalDynamicControls% * 25
Properties.Width=CtrlWidth%
Properties.Height=200
Properties.TAB=102
Properties.ForeColor=COL_NEUTRAL%
Properties.BackColor=COL_BACKGROUND%
Properties.FontName="MS Sans Serif"
Properties.FontSize=8
Properties.FontSels=GFS_BOLD%
Properties.ToolTip="Dynamic Listbox"
Properties.Attributes = DCA_LB_HORIZSCROLLBAR + DCA_LB_BELOW
Properties.SEPARATORS = " '|', #7 "
Properties.TABULATIONS ="0,100,100"
Properties.Text="Nom|Prenom"
j% =DC_CreateControl% (Properties)
insert at END "Jean|Dupont" to Properties.Ctrl
dispose @Properties
```

Constantes DCA_MLE_* (Librairie NSDYNCTRL)

Ces constantes permettent d'implémenter des attributs de contrôles dynamiques (Dynamic Control Attributes) pour le contrôle MLE.

Syntaxe	Déclaration interne	Description
DCA_MLE_LOCKTEXT	1	Permet de verrouiller la saisie.

DCA_MLE_DISABLE	2	Permet de désactiver le contrôle.
DCA_MLE_UPCASE	4	Permet de traduire automatiquement en majuscules les caractères saisis en minuscules.
DCA_MLE_ACTIVETAB	8	Permet de forcer l'utilisation de la touche [Tab] au passage du focus au champ suivant. Lorsqu'elle est non cochée (par défaut), la touche Tab permet de déplacer le curseur à l'intérieur de la MLE sur la position de tabulation suivante dont la largeur est spécifiée dans Tab stop interval.
DCA_MLE_OVER	16	Permet d'afficher le contrôle en aspect bombé.
DCA_MLE_BELOW	32	Permet d'afficher le contrôle en aspect creux.
DCA_MLE_WORDWRAP	64	Permet de retourner automatiquement à la ligne.
DCA_MLE_NOADJUSTPOS	128	Permet de ne pas ajuster la hauteur de la liste.
DCA_MLE_HIDDEN	8388608	Permet de cacher le contrôle

Constantes DCA_CB_* (Librairie NSDYNCTRL)

Ces constantes permettent d'implémenter des attributs de contrôles dynamiques (Dynamic Control Attributes) pour le contrôle Combo Box.

Syntaxe	Déclaration interne	Description
DCA_CB_DISABLE	1	Désactive le contrôle. Détermine l'état du contrôle au démarrage de la boîte de dialogue à laquelle elle appartient.
DCA_CB_OVER	16	Permet d'afficher le contrôle en aspect bombé.

DCA_CB_BELOW	32	Permet d'afficher le contrôle en aspect creux.
DCA_CB_HIDDEN	8388608	Permet de cacher les contrôles.

Constantes DCA_CHK_* (Librairie NSDYNCTRL)

Ces constantes permettent d'implémenter des attributs de contrôles dynamiques (Dynamic Control Attributes) pour le contrôle Check Box.

Syntaxe	Déclaration interne	Description
DCA_CHK_AUTOSIZE	1	Permet d'adapter la taille de la Check Box au texte affiché.
DCA_CHK_DISABLE	2	Permet de désactiver le contrôle.
DCA_CHK_3STATES	4	Permet de définir une CheckBox à 3 états. La majorité des Check Boxes sont à 2 états : coché ou non coché. Il est possible d'obtenir une Check Box à 3 états (coché, non coché, grisé).
DCA_CHK_OVER	8	Permet d'afficher le contrôle en aspect bombé.
DCA_CHK_BELOW	16	Permet d'afficher le contrôle en aspect creux.
DCA_CHK_HIDDEN	8388608	Permet de cacher le contrôle.

Constantes DCA_RB_* (Librairie NSDYNCTRL)

Ces constantes permettent d'implémenter des attributs de contrôles dynamiques (Dynamic Control Attributes) pour le contrôle Radio Button.

Syntaxe	Déclaration interne	Description
DCA_RB_DISABLE	1	Permet de désactiver le contrôle.
DCA_RB_AUTOSIZE	2	Permet d'adapter la taille du Radio Button au texte affiché.

DCA_RB_OVER	4	Permet d'afficher le contrôle en aspect bombé.
DCA_RB_BELOW	8	Permet d'afficher le contrôle en aspect creux.
DCA_RB_HIDDEN	8388608	Permet de cacher le contrôle.

Constantes DCA_ST_* (Librairie NSDYNCTRL)

Ces constantes permettent d'implémenter des attributs de contrôles dynamiques (Dynamic Control Attributes) pour le contrôle Static Text.

Syntaxe	Déclaration interne	Description
DCA_ST_CENTER	1	Permet de justifier le texte au centre
DCA_ST_LEFT	2	Permet de justifier le texte à gauche
DCA_ST_RIGHT	4	Permet de justifier le texte à droite
DCA_ST_HALFTONE	8	Permet de griser le texte d'une zone de texte.
DCA_ST_MNEMONIC	16	Permet d'assigner un raccourci clavier à un contrôle Static Text. Par exemple, pour un contrôle ~Exit, le raccourci [Alt]+E va permettre de positionner le focus sur le contrôle qui lui est associé.
DCA_ST_ERASERECT	32	Permet d'effacer le fond rectangulaire derrière une zone de texte
DCA_ST_WORDWRAP	64	Permet de définir une zone de texte sur plusieurs lignes
DCA_ST_AUTOSIZE	128	Permet de rendre visible tout le texte
DCA_ST_MARGIN	256	Permet d'encadrer le contrôle Static Text.
DCA_ST_OVER	512	Permet d'afficher le contrôle en aspect bombé.
DCA_ST_BELOW	1024	Permet d'afficher le contrôle en aspect creux.

DCA_ST_HIDDEN	8388608	Permet de cacher le contrôle
---------------	---------	------------------------------

Constantes DCA_BMP_* (Librairie NSDYNCTRL)

Ces constantes permettent d'implémenter des attributs de contrôles dynamiques (Dynamic Control Attributes) pour le contrôle Bitmap.

Syntaxe	Déclaration interne	Description
DCA_BMP_ADJUSTSIZE	1	Pour ajuster la taille d'une bitmap
DCA_BMP_DISABLE	2	Pour désactiver le contrôle
DCA_BMP_ICON	4	Pour associer un comportement du contrôle Icon à une Bitmap
DCA_BMP_PUSHBUTTON	8	Pour associer un comportement de Push Button à une Bitmap
DCA_BMP_CHECKBOX	16	Pour associer un comportement de Check Box à une Bitmap
DCA_BMP_NOCHECKING	32	Pour interdire un comportement de Check Box à une Bitmap
DCA_BMP_HIDDEN	8388608	Pour cacher le contrôle

Exemple :

```
Local DC_ControlProperties@ Properties
LOCAL i%, j%, h%, h1%, h2%
SEND EXECUTED TO PB_DESTROY
new @Properties
Properties.DialogHandle=self%
Properties.ID =Firstid% + GetTotalDynamicControls%
Properties.X=4
Properties.Y=getClientheight% - 25 - GetTotalDynamicControls% * 25
Properties.Kind=DI_BITMAP%
Properties.Tab =102
Properties.Width=32
Properties.Height=32
h% = createbmp%("(NS-BMP)\ADD.BMP")
h1% = createbmp%("(NS-BMP)\CLOSE.BMP")
h2% = createbmp%("(NS-BMP)\CANCEL.BMP")
Properties.disabled=':&h%
Properties.pressed=':&h1%
Properties.released=':&h2%
Properties.Attributes = DCA_BMP_PUSHBUTTON+DCA_BMP_ADJUSTSIZE
j% =DC_CreateControl% (Properties)
Properties.CTRL.FORECOLOR=TRANSP_TOPLEFT%
Properties.CTRL.Relief=2
dispose @Properties
```

Voir aussi [DC_ControlProperties](#)

Constantes DCA_GB_* (Librairie NSDYNCTRL)

Ces constantes permettent d'implémenter des attributs de contrôles dynamiques (Dynamic Control Attributes) pour le contrôle Group Box.

Syntaxe	Déclaration interne	Description
DCA_GB_CENTER	1	Permet de justifier le texte au centre.
DCA_GB_LEFT	2	Permet de justifier le texte à gauche.
DCA_GB_RIGHT	4	Permet de justifier le texte à droite.
DCA_GB_HALFTONE	8	Permet de griser le texte d'une zone de texte
DCA_GB_MNEMONIC	16	Permet d'assigner un raccourci clavier à un contrôle Group Box.
DCA_GB_ERASERECT	32	Permet d'effacer le fond rectangulaire derrière une zone de texte
DCA_GB_OVER	64	Permet d'afficher le contrôle en aspect bombé.
DCA_GB_BELOW	128	Permet d'afficher le contrôle en aspect creux.
DCA_GB_OVER1	256	Permet d'afficher le contrôle en aspect bombé sans texte. Attention, le texte ne sera plus visible.
DCA_GB_BELOW1	512	Permet d'afficher le contrôle en aspect creux sans texte. Attention, le texte ne sera plus visible.
DCA_GB_HIDDEN	8388608	Permet de cacher le contrôle.

Constantes DCA_VS_* (Librairie NSDYNCTRL)

Ces constantes permettent d'implémenter des attributs de contrôles dynamiques (Dynamic Control Attributes) pour le contrôle Vscroll (barre de défilement verticale).

Syntaxe	Déclaration interne	Description
DCA_VS_DISABLE	1	Permet de désactiver le contrôle.
DCA_VS_HIDDEN	8388608	Permet de cacher le contrôle.

Constantes DCA_HS_* (Librairie NSDYNCTRL)

Ces constantes permettent d'implémenter des attributs de contrôles dynamiques (Dynamic Control Attributes) pour le contrôle Hscroll (barre de défilement horizontale).

Syntaxe	Déclaration interne	Description
DCA_HS_DISABLE	1	Permet de désactiver le contrôle.
DCA_HS_HIDDEN	8388608	Permet de cacher le contrôle.

Constantes DCA_MNU_* (Librairie NSDYNCTRL)

Ces constantes permettent d'implémenter des attributs de contrôles dynamiques (Dynamic Control Attributes) pour le contrôle Menu.

Syntaxe	Déclaration interne	Description
DCA_MNU_RIGHTSIDE	1	Permet d'afficher un item de menu à droite de la barre de menu.
DCA_MNU_SEPARATOR	2	Permet d'afficher une barre de séparation après une option de menu
DCA_MNU_DISABLED	4	Permet de désactiver le contrôle
DCA_MNU_CHECKED	8	Permet de cocher une option de menu dès l'ouverture de la fenêtre
DCA_MNU_SUBMENU	16	Permet de définir un sous-menu
DCA_MNU_HIDDEN	8388608	Permet de cacher le contrôle.

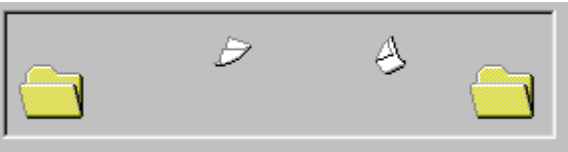

Constante DI_WINCMNCTRL% (Librairie NSDYNCTRL)

Constante indiquant la valeur d'un contrôle de type Common Control.

Lorsque le paramètre dynamique .KIND=DI_WINCMNCTRL, le sous-type (.SUBKIND) correspond à une des constantes suivantes :

- DI_WCC_Animation%
- DI_WCC_DateTimePicker%
- DI_WCC_HotKey%
- DI_WCC_ListView%
- DI_WCC_MonthCalendar%
- DI_WCC_ProgressBar%
- DI_WCC_TrackBar%
- DI_WCC_TreeView%

Les constantes suivantes correspondent aux styles communs à tous les Windows Common Controls :

Syntaxe	Déclaration interne	Description
WCC_Transparent	1	Permet de rendre la couleur d'arrière-plan du contrôle transparente.
WCC_ModalFrame	32	Dessine la bordure du contrôle en relief.
WCC_ClientEdge	512	Dessine la bordure du contrôle en relief creux. 
WCC_StaticEdge	131072	Dessine la bordure du contrôle en relief creux. 
WCC_HIDDEN	8388608	Permet de cacher le contrôle

Exemple :

```
Local DC_ControlProperties@ Properties
LOCAL i%, j%, h%, h1%, h2%
SEND EXECUTED TO PB_DESTROY
new @Properties
Properties.Kind=DI_WINCMNCTRL%
Properties.Subkind =DI_WCC_DateTimePicker%
Properties.Style =DTS_LONGDATEFORMAT
Properties.DialogHandle=self%
```

```

Properties.ID =Firstid% + GetTotalDynamicControls%
Properties.X=4
Properties.Y=getClientheight% - Y%
Properties.Width=CtrlWidth%
Properties.Height=20
Properties.ForeColor=COL_NEUTRAL%
Properties.BackColor=COL_BACKGROUND%
Properties.FontName="MS Sans Serif"
Properties.FontSize=8
Properties.FontSels=GFS_BOLD%
Properties.ToolTip="Dynamic Date Time Picker"
j% =DC_CreateControl% (Properties)
dispose @Properties

```

Constante DI_WCC_Animation% (Librairie NSDYNCTRL)

Constante indiquant un contrôle de type Animation.

Les constantes suivantes associées à la constante DI_WCC_Animation% permettent de paramétrer le contrôle Animation :

Syntaxe	Déclaration interne	Description
ACS_AUTOPLAY	\$0004	Permet d'exécuter automatiquement l'animation.
ACS_TIMER	\$0008	Permet de régler le nombre de secondes pendant lequel l'animation doit s'exécuter.
ACS_CENTER	\$0001	Permet de centrer l'animation dans la fenêtre du contrôle Animation.
ACS_TRANSPARENT	\$0002	Permet d'agréger la couleur de fond du contrôle avec la couleur principale de la fenêtre.

Constante DI_WCC_DateTimePicker% (Librairie NSDYNCTRL)

Constante indiquant un contrôle de type Date Time Picker.

Les constantes suivantes associées à la constante DI_WCC_DateTimePicker% permettent de paramétrer le contrôle Date Time Picker :

Syntaxe	Déclaration interne	Description
---------	---------------------	-------------

DTS_SHORTDATEFORMAT	0	Permet de spécifier un format de date court. (ex. : « 26/03/01 »)
DTS_LONGDATEFORMAT	4	Permet de spécifier un format de date long. (ex. : « Mardi 27 Mars 2001 »)
DTS_TIMEFORMAT	8	Permet de spécifier un format d'heure (ex. : « 14:53:18 »).
DTS_UPDOWN	1	Permet de définir un contrôle représenté par une paire de flèches permettant de retirer ou d'ajouter une valeur dans le contrôle Date associé. Ce style peut être utilisé à la place du calendrier qui est l'option par défaut.
DTS_SHOWNONE	2	Permet de n'afficher aucune sélection de date dans le contrôle. Ce style affiche une case à cocher dans le contrôle permettant de valider une date saisie ou sélectionnée.
DTS_SHORTDATECENTURYFORMAT	12	Permet de spécifier l'année sur quatre chiffres (ex. : « 26/03/2001 »)
DTS_APPCANPARSE	16	Permet à l'utilisateur d'analyser les entrées et d'effectuer les actions nécessaires. Il autorise les utilisateurs à éditer dans l'aire client du contrôle quand ils activent la touche [F2].

DTS_RIGHTALIGN	32	Permet l'alignement à droite (par défaut, le contrôle Date and Time Picker est aligné à gauche).
----------------	----	--

Constante DI_WCC_Hotkey% (Librairie NSDYNCTRL)

Constante indiquant un contrôle de type HotKey.

Constante DI_WCC_ListView% (Librairie NSDYNCTRL)

Constante indiquant un contrôle de type HotKey.

Les constantes suivantes associées à la constante DI_WCC_ListView% permettent de paramétrer le contrôle List View :

Syntaxe	Déclaration interne	Description
LVS_ICON	\$0000	Permet d'afficher les données sous forme de grandes icônes.
LVS_REPORT	\$0001	Permet d'afficher les données sous forme détaillée.
LVS_SMALLICON	\$0002	Permet d'afficher les données sous forme de petites icônes.
LVS_LIST	\$0003	Permet d'afficher les données sous forme de liste.
LVS_TYPEMASK	\$0003	Permet de spécifier le type de masque d'une liste d'image.
LVS_TYPESTYLEMASK	\$fc00	Permet de spécifier le style du type de masque d'une liste d'image.
LVS_ALIGNMASK	\$0c00	Permet d'aligner le masque d'une liste d'image
LVS_SINGLESEL	\$0004	Permet de restreindre la sélection à un seul élément.

LVS_SHOWSELALWAYS	\$0008	Permet de rendre la sélection toujours visible, même si le contrôle n'a plus le focus.
LVS_SORTASCENDING	\$0010	Permet de trier les données par taille ascendante.
LVS_SORTDESCENDING	\$0020	Permet de trier les données par taille descendante
LVS_SHAREIMAGELISTS	\$0040	Permet de ne pas supprimer la liste d'images (Image list) en cas de destruction du contrôle. Ce style permet l'utilisation d'une même liste d'images pour plusieurs contrôles List View.
LVS_NOLABELWRAP	\$0080	Permet au contrôle d'afficher le texte de l'élément sur une ligne unique en vue icônes.
LVS_AUTOARRANGE	\$0100	Permet au contrôle de maintenir une vue par icônes.
LVS_EDITLABELS	\$0200	Permet au contrôle d'éditer le texte de l'élément en place.
LVS_OWNERDATA	\$1000	Permet de spécifier un contrôle List View virtuel. Ce style permet au contrôle de manipuler des millions d'éléments.
LVS_NOSCROLL	\$2000	Permet de désactiver la barre de défilement sur le contrôle. Ce style n'est pas compatible avec les vues par liste ou détaillée.
LVS_ALIGNTOP	\$0000	Permet d'afficher les éléments alignés en haut de la liste en vue icône (grande ou petite).
LVS_ALIGNLEFT	\$0800	Permet d'afficher les éléments alignés à gauche de la liste en vue icône (grande ou petite).

LVS_OWNERDRAWFIXED	\$0400	Le propriétaire du contrôle peut dessiner des éléments dans une vue détaillée. Le contrôle ListView envoie un message WM_DRAWITEM pour peindre chaque élément. Il n'envoie pas des messages séparés pour chacun des sous-éléments.
LVS_NOCOLUMNHEADER	\$4000	Permet de désactiver l'affichage des en-têtes des colonnes dans une vue détaillée.
LVS_NOSORTHEADER	\$8000	Les en-têtes des colonnes ne sont pas cliquables et ne permettent donc pas le tri par colonne dans une vue détaillée.

Constante **DI_WCC_MonthCalendar%** (Librairie NSDYNCTRL)

Constante indiquant un contrôle de type Month Calendar.

Les constantes suivantes associées à la constante DI_WCC_MonthCalendar% permettent de paramétrer le contrôle Month Calendar :

Syntaxe	Déclaration interne	Description
MCS_DAYSTATE	1	Affiche certaines dates en gras en fonction des messages NOTIFY MCN_GETDAYSTATE% envoyés par le contrôle.
MCS_MULTISELECT	2	Permet de sélectionner un intervalle de dates à l'intérieur du contrôle. Par défaut, l'intervalle maximum est d'une semaine.
MCS_WEEKNUMBERS	4	Permet de spécifier le le numéro de la semaine (1-52) à gauche de chaque ligne.
MCS_NOTODAYCIRCLE	8	Permet de ne pas encercler la date du jour.

MCS_NOTODAY	16	Permet de ne pas afficher la date du jour en bas du contrôle.
-------------	----	---

Constante DI_WCC_ProgressBar% (Librairie NSDYNCTRL)

Constante indiquant un contrôle de type Month Calendar.

Les constantes suivantes associées à la constante DI_WCC_ProgressBar% permettent de paramétrer le contrôle Progress Bar :

Syntaxe	Déclaration interne	Description
PBS_SMOOTH	\$01	Affiche une barre de progression pleine
PBS_VERTICAL	\$04	Affiche une barre de progression dont l'axe est vertical

Constante DI_WCC_TrackBar% (Librairie NSDYNCTRL)

Constante indiquant un contrôle de type Track Bar.

Les constantes suivantes associées à la constante DI_WCC_TrackBar% permettent de paramétrer le contrôle Track Bar :

Syntaxe	Déclaration interne	Description
TBS_AUTOTICKS	\$0001	Permet d'afficher une marque de graduation pour chaque valeur.
TBS_VERT	\$0002	Permet d'orienter le contrôle Track Bar verticalement.
TBS_HORZ	\$0000	Permet d'orienter le contrôle Track Bar horizontalement. C'est l'orientation par défaut.
TBS_TOP	\$0004	Permet de situer les marques de graduation en haut.
TBS_BOTTOM	\$0000	Permet de situer les marques de graduation en bas.

TBS_LEFT	\$0004	Permet de situer les marques de graduation à gauche.
TBS_RIGHT	\$0000	Permet de situer les marques de graduation à droite.
TBS_BOTH	\$0008	Permet de situer les marques de graduation en haut et en bas.
TBS_NOTICKS	\$0010	Permet de n'afficher aucune marque de graduation.
TBS_ENABLESELRANGE	\$0020	Permet d'afficher une sélection unique.
TBS_FIXEDLENGTH	\$0040	Permet de fixer une largeur.
TBS_NOTHUMB	\$0080	Permet de ne pas afficher de curseur.
TBS_TOOLTIPS	\$0100	Permet de créer automatiquement une info-bulle affichant la position courante du curseur.
TBS_REVERSED	\$0200	Permet de sélectionner la plus petite valeur (la plupart du temps correspond à la valeur minimale) comme valeur haute et la plus grande valeur (correspondant en général à la valeur maximale) comme valeur basse.
TBS_DOWNSIZELEFT	\$0400	Permet de définir le bas à gauche et le haut à droite (par défaut, le positionnement du bas est à droite et le haut à gauche).

Constante **DI_WCC_TreeView%** (Librairie NSDYNCTRL)

Constante indiquant un contrôle de type Tree View.

Les constantes suivantes associées à la constante DI_WCC_TreeView% permettent de paramétrer le contrôle Tree View :

Syntaxe	Déclaration interne	Description
---------	---------------------	-------------

TVS_HASBUTTONS	\$0001	Permet d'afficher un bouton à gauche (+) ou (-) à côté des éléments parents de la liste. Ils permettent de développer ou de réduire les éléments enfants.
TVS_HASLINES	\$0002	Permet d'utiliser des lignes pour permettre une meilleure visualisation de la hiérarchie entre éléments enfants et parents.
TVS_LINESATROOT	\$0004	Permet d'utiliser des lignes verticales pour relier les éléments à l'élément racine de la liste. Ce style n'est effectif que si la constante TVS_HASLINES est également spécifié.
TVS_EDITLABELS	\$0008	Permet d'éditer et de modifier les propriétés de l'étiquette de l'élément sélectionné.
TVS_DISABLEDROGDROP	\$0010	Permet de désactiver l'option glisser/déposer.
TVS_SHOWSELALWAYS	\$0020	Permet de toujours afficher les items.
TVS_RTLREADING	\$0040	Permet de spécifier l'affichage pour une lecture de droite à gauche.
TVS_NOTOOLTIPS	\$0080	Permet de désactiver les info-bulles.
TVS_CHECKBOXES	\$0100	Permet d'activer des cases à cocher pour les éléments du contrôle Tree View. Une case à cocher est affichée uniquement si une image est associée à l'élément.
TVS_TRACKSELECT	\$0200	Permet d'activer l'affichage en surbrillance des nœuds pointés par la souris.
TVS_SINGLEEXPAND	\$0400	Entraîne l'extension de l'élément sélectionné et plie tous les autres.
TVS_INFOTIP	\$0800	Permet de récupérer l'information de l'info-bulle.

TVS_FULLROWSELECT	\$1000	Permet de sélectionner la ligne complète. Active la sélection complète de la ligne dans une vue arborescente. La ligne complète de l'élément sélectionné est surligné quel que soit l'élément de la ligne sélectionné. Ce style ne peut être utilisé avec la constante STV_HASLINES.
TVS_NONEVENHEIGHT	\$4000	Permet de fixer la hauteur des éléments à une hauteur impaire avec le message TVM_SETITEMHEIGHT%. Par défaut, la hauteur des éléments a une valeur paire.
TVS_NOHSCROLL	\$8000	Permet de désactiver le défilement horizontal.
TVS_NOSCROLL	\$2000	Permet de désactiver le défilement horizontal et vertical. Le contrôle n'affiche aucune barre de défilement.

SEGMENT DC_ControlProperties (Librairie NSDYNCTRL)

Le segment DC_ControlProperties est passé en paramètre aux différentes fonctions DC_*. Il permet de créer des contrôles dynamiques selon les propriétés positionnées.

Syntaxe	SEGMENT DC_ControlProperties POINTER DialogHandle INTEGER Kind INTEGER ID INTEGER X INTEGER Y INTEGER Width INTEGER Height INTEGER Attributes INTEGER ForeColor INTEGER BackColor CSTRING FontName
----------------	---

	CSTRING FontSize CSTRING FontSels CSTRING Anchor CSTRING ToolTip INTEGER Tab INTEGER VALUE CONTROL Ctrl POINTER CtrlHandle INTEGER ParentID INTEGER Picture INT(4) MaxLen CSTRING Characters CSTRING Format CSTRING TEXT CSTRING SEPARATORS CSTRING TABULATIONS INT(2) TabStopInterval INT(1) NBLINES INTEGER GROUPID CSTRING Released CSTRING Pressed CSTRING Disabled INTEGER SubKind INTEGER style INTEGER exstyle INT(1) Version INT(2) Internal% INTEGER Internal2% POINTER Data CSTRING Paramstr ENDSEGMENT		
Champs	DialogHandle	POINTER	NSHandle du parentWindow (self%)
	Kind	INTEGER	une constante DI_* (librairie NsMisc)
	ID	INTEGER	identifiant du contrôle
	X	INTEGER	coordonnée X

Y	INTEGER	coordonnée Y
Width	INTEGER	largeur (sauf pour icône et barre de défilement verticale)
Height	INTEGER	hauteur pour Push Buttons, ListBox, ComboBox, Combo Entry, MLE, GroupBox, Static Text, Vertical Scrollbar, Bitmaps
Attributes	INTEGER	une constante DCA_* Disabled Autosize etc
ForeColor	INTEGER	couleur d'avant-plan (ne fonctionne pas pour les contrôles Icon et Bitmap)
BackColor	INTEGER	couleur d'arrière-plan (ne fonctionne pas pour les contrôles Icon et Bitmap)
FontName	CSTRING	nom de la police (ne fonctionne pas pour les contrôles Icon et Bitmap)
FontSize	CSTRING	taille de la police (ne fonctionne pas pour les contrôles Icon et Bitmap)
FontSels	CSTRING	constante GFS_*(librairie NSGraph). Ne fonctionne pas pour les contrôles Icon et Bitmap
Anchor	CSTRING	une constante NS_AS_* (librairie NSMisc)
ToolTip	CSTRING	bulle d'aide (ne fonctionne pas pour les contrôles Icon)
Tab	INTEGER	dernier contrôle dans la séquence de tabulation ou l'ID du contrôle avec lequel le Static text ou le Group Box est relié. Si vous mentionnez un contrôle dans ce paramètre, faire attention à son existence réelle
VALUE	INTEGER	valeur du RadioButton, CheckBox, Icon. Sélectionner une constante SPTR_*
Ctrl	CONTROL	contrôle créé
CtrlHandle	POINTER	handle du contrôle créé

ParentID	INTEGER	Identifiant du MenuItem Parent, Menuitem ou 0
Picture	INTEGER	une constante SMIP_ *% (librairie NSMisc)
MaxLen	INT(4)	largeur maximale d'un contrôle EntryField (par défaut, la largeur maximale est de 31) ou nombre maximum de caractères dans un contrôle MLE
Characters	CSTRING	caractères autorisés
Format	CSTRING	format à afficher
TEXT	CSTRING	label du contrôle
SEPARATORS	CSTRING	chaîne de caractères avec des virgules séparant chaque caractère. Ex. : "' ', #7"
TABULATIONS	CSTRING	virgule séparant les colonnes. Ex. : "0, 15,100, 100"
TabStopInterval	INT(2)	intervalle de tabulation dans un contrôle MLE
NBLINES	INT(1)	taille d'une liste déroulante en nombre de lignes pour un contrôle ComboBox
GROUPID	INTEGER	identifiant d'un contrôle Group pour des RadioButtons
Released	CSTRING	chemin complet d'une bitmap. Utilisation possible de variables d'environnement
Pressed	CSTRING	chemin complet d'une bitmap correspondant au bouton souris enfoncé. Utilisation possible de variables d'environnement
Disabled	CSTRING	chemin complet d'une bitmap correspondant au bouton souris désactivé. Utilisation possible de variables d'environnement
SubKind	INTEGER	une constante DI_WCC_*

	Style	INTEGER	une constante XXS_*. Les caractères X correspondent aux initiales du contrôle.
	Exstyle	INTEGER	une constante WCC_*

Exemple :

```
; Création d'un contrôle ComboBox
Local DC_ControlProperties@ Properties
LOCAL i%, j%, h%, h1%, h2%
SEND EXECUTED TO PB_DESTROY
new @Properties
Properties.Kind=DI_COMBBOX%
Properties.DialogHandle=self%
Properties.ID =Firstid% + GetTotalDynamicControls%
Properties.X=4
Properties.Y=getClientheight% - y%
Properties.Width=CtrlWidth%
Properties.Height=24
y%=y% + Properties.Height + 4
Properties.TAB=102
Properties.ForeColor=COL_NEUTRAL%
Properties.BackColor=COL_BACKGROUND%
Properties.FontName="MS Sans Serif"
Properties.FontSize=8
Properties.FontSels=GFS_BOLD%
Properties.ToolTip="Dynamic Combobox"
Properties.Attributes = DCA_CB_BELOW
Properties.SEPARATORS = " '|', #7 "
Properties.TABULATIONS ="0,100,100"
j% =DC_CreateControl% (Properties)
insert at END "Jean|Dupont" to Properties.Ctrl
dispose @Properties
```

Fonction ItExists% (Librairie NSDYNCTRL)

Permet d'identifier la présence d'un contrôle MenuItem.

Syntaxe	ItExists% (Handle, ID)		
Paramètres	Handle	POINTER	handle d'un contrôle
	ID	INTEGER	identifiant du contrôle
Valeur retournée	INTEGER 0, si le contrôle est un MenuItem, non valide ou n'existe pas, autrement renvoie une valeur différente de 0.		

Fonction GetTotalDynamicControls% (Librairie NSDYNCTRL)

Retourne le nombre total de contrôles dynamiques.

Syntaxe	GetTotalDynamicControls%
Valeur retournée	INTEGER

Exemple :

```

Local DC_ControlProperties@ Properties
LOCAL i%, j%, h%, h1%, h2%
SEND EXECUTED TO PB_DESTROY
new @Properties
Properties.Kind=DI_ICON%
Properties.DialogHandle=self%
Properties.ID =Firstid% + GetTotalDynamicControls%
Properties.GroupID =Properties.ID
Properties.X=4
Properties.Y=getClientheight% - y%
Properties.Height=24
Properties.Width=40
y%=y% + Properties.Height + 4
Properties.ForeColor=COL_NEUTRAL%
Properties.BackColor=COL_BACKGROUND%
Properties.Anchor=NS_AS_NONE
Properties.ToolTip="Dynamic Icon"
Properties.Value = SPTR_ARROW%
; be carefull if tab <> 0: the next control in the tab sequence
; should be created otherwise it'll hangs
Properties.Tab = 102
j% =DC_CreateControl% (Properties)
Properties.Kind=DI_ICON%
Properties.DialogHandle=self%
Properties.ID =Firstid% + GetTotalDynamicControls%
Properties.X=Properties.X+4+Properties.Width
Properties.Width=40
Properties.Anchor=NS_AS_NONE
Properties.ToolTip="Dynamic Icon"
Properties.Value = SPTR_ILLEGAL%
Properties.Tab = 103
j% =DC_CreateControl% (Properties)
dispose @Properties

```

Fonction **DC_CreateControl%** (Librairie NSDYNCTRL)

Crée un contrôle dynamique qui retourne le nombre total de contrôles dynamiques.

Syntaxe	DC_CreateControl% (DC_ControlProperties)		
Paramètre	DC_ControlProperties	SEGMENT	segment DC_ControlProperties
Valeur retournée	INTEGER		

Voir aussi Segment DC_ControlProperties

Fonction **DC_DestroyControl%** (Librairie NSDYNCTRL)

Supprime un contrôle dynamique qui retourne le nombre total de contrôles dynamiques.

Syntaxe	DC_DestroyControl% (<i>HWnd, Id%</i>)		
Paramètres	HWnd	POINTER	handle de fenêtre
	Id%	INT	identifiant du contrôle à supprimer
Valeur retournée	INTEGER		

Voir aussi Segment [DC_ControlProperties](#)

Instruction **DC_GetControlPropertiesFromID** ([Librairie NSDYNCTRL](#))

Permet de récupérer les propriétés d'un contrôle.

Syntaxe	DC_GetControlPropertiesFromID <i>ControlProperties</i>		
Paramètre	ControlProperties	SEGMENT	segment DC_ControlProperties

Avant d'appeler l'instruction DC_GetControlPropertiesFromID, il est nécessaire d'alimenter les champs DialogHandle et ID du segment DC_ControlProperties.

Exemple :

```
Global POINTER HCALLBACK
Global Firstid%
SEGMENT Exchange
INTEGER MSG
POINTER PARM1
POINTER PARM2
DC_ControlProperties@ Properties
ENDSEGMENT
Function OldWndProc(POINTER HWND,INTEGER ID,INTEGER MSG,POINTER PARM1,POINTER PARM2)
RETURN INTEGER DYNAMIC
Function NewWndProc(POINTER HWND,INTEGER ID,INTEGER MSG,POINTER PARM1,POINTER PARM2)
RETURN INTEGER
local DC_ControlProperties@ Properties
local Exchange@ pExchange
EVALUATE MSG
    WHERE EVENT_EXECUTED%,EVENT_SELECTED%
        if ID >= FIRSTID%
            new @Properties
            Properties.DialogHandle =HWND
                                Properties.ID =ID
            DC_GetControlPropertiesFromID Properties
            new @pExchange
            pExchange.MSG = MSG
            pExchange.PARM1 = PARM1
            pExchange.PARM2 = PARM2
            @pExchange.Properties = Properties
            Send USER1, @pExchange, MSG to HWND
            dispose @Properties
            dispose @pExchange
        else
            RETURN CALL_PREVCALLBACK(hCallback, HWND, id, msg, parm1, parm2)
        endif
```

```
ENDWHERE  
ELSE  
    RETURN CALL_PREVCALLBACK(hCallback, HWND, id, msg, parm1, parm2)  
ENDEVALUATE  
EndFunction  
Firstid% = 1000
```

Voir aussi Segment [DC_ControlProperties](#)

Instruction **DC_GetControlProperties** ([Librairie NSDYNCTRL](#))

Permet de récupérer les propriétés d'un contrôle.

Syntaxe	DC_GetControlProperties <i>Handle, ControlProperties</i>		
Paramètres	Handle	POINTER	handle du contrôle
	ControlProperties	SEGMENT	segment DC_ControlProperties

Voir aussi Segment [DC_ControlProperties](#)

LIBRAIRIE NSMAPS

Cette librairie permet de gérer une MAP de pointeur ou de String dynamiques

Une map est utilisable comme un tableau, à la différence que l'indexation se fait à l'aide d'une string, et non pas d'un entier.

Elle peut être utilisée par, par exemple, pour mémoriser une liste de personnes, indexée par leur N° INSEE

Référence de la librairie NSMAPS

Référence de la librairie NSMAPS

Voici la liste des instructions et fonctions de la librairie NSMAP

.

Création de map

Création d'une map de pointeurs

function NSMAPS MAP_NEW

Création d'une map de pointeurs avec désallocation automatique

function NSMAPS MAP_NEW2

Gestion des éléments

Stockage d'un élément pointeur

instruction NSMAPS MAP_PUT

Récupération d'un élément pointeur

function NSMAPS MAP_GET

Map de chaînes dynamiques (DynStr)

Création d'une map de chaînes

function NSMAPS MAP_STRING_NEW

Récupération d'une chaîne

function NSMAPS MAPGET_STRING

Stockage d'une chaîne

instruction NSMAPS MAP_PUT_STRING

Test si une clé est présente dans la map

function NSMAPS MAP CONTAINS%

Retire une clé et la valeur associée de la map

instruction NSMAPS MAP REMOVE

Vidage de la MAP

Instruction NSMAPS_MAP_CLEAR

Libération de la map

instruction NSMAPS MAP DISPOSE

Nombre d'Elements de la map

function NSMAPS MAP SIZE

Gestion des itérateurs

Création d'un itérateur.

function NSMAPS MAP GET ITERATOR

Essaye de passer a l'élément suivant,

function NSMAPS MAP ITERATOR NEXT

Valeur de l'élément courant

function NSMAPS MAP ITERATOR GETVALUE

Valeur de l'élément courant

function NSMAPS MAP ITERATOR GETSTRING

Clé de l'élément courant

function NSMAPS MAP ITERATOR GETKEY

Libération de l'itérateur

instruction NSMAPS MAP ITERATOR DISPOSE

Instruction NSMAPS_MAP_REMOVE (Librairie NSMAPS)

Enlève un objet de la map.

Syntaxe	NSMAPS_MAP_REMOVE (<i>hMap, key</i>)			
Paramètres	hMap	pointer	l	handle de la map

	key	cstring	I	clé de l'objet à supprimer de la map
--	-----	---------	---	--------------------------------------

Remarque

Si la map a été allouée par NSMAPS_MAP_NEW2, l'appel automatique à l'instruction de désallocation est effectuée pour le data

Voir aussi

NSMAPS_MAP_PUT, NSMAPS_MAP_PUT_STRING

Fonction NSMAPS_MAP_ITERATOR_GETVALUE (Librairie NSMAPS)

Retrouve un objet depuis une Map, à partir de la position courante de l'itérateur.

Syntaxe	NSMAPS_MAP_ITERATOR_GETVALUE (<i>iterator</i>)			
Paramètres	iterator	pointer	I	itérateur parcourant la map
Valeur retournée	POINTER Pointeur sur l'objet trouvé			

Exemple :

```

segment SEG_CLIENT
  nom$
  prenom$
  age%
endSegment

local pointer iterator
; boucle avec itérateur

trace "--- boucle de parcours avec itérateur"
iterator = NSMAPS_MAP_GET_ITERATOR(mymap)
repeat
  @pCli = SEG_CLIENT(NSMAPS_MAP_ITERATOR_GETVALUE(iterator))
  trace "boucle Itérateur" && NSMAPS_MAP_ITERATOR_GETKEY(iterator) && ":" && pCli.nom$ &
  '/' & pCli.prenom$ & '/' & pCli.age%
until NOT NSMAPS_MAP_ITERATOR_NEXT(iterator)

; Libération de l'itérateur
NSMAPS_MAP_ITERATOR_DISPOSE iterator

```

Voir aussi

[NSMAPS_MAP_ITERATOR_GETSTRING](#), [NSMAPS_MAP_ITERATOR_GETKEY](#)

Fonction [NSMAPS_MAP_ITERATOR_GETSTRING](#) (Librairie [NSMAPS](#))

Retrouve une string depuis une Map de string, à partir de la position courante de l'itérateur.

Syntaxe	NSMAPS_MAP_ITERATOR_GETSTRING (<i>iterator</i>)		
Paramètres	iterator	pointer	I itérateur parcourant la map
Valeur retournée	dynStr string stockée à la position courante de la map		

Exemple :

```
local pointer iterator
; boucle avec itérateur

trace "--- boucle de parcours avec itérateur"
iterator = NSMAPS_MAP_GET_ITERATOR(mymap)
repeat
    trace ("boucle Iterator" && NSMAPS_MAP_ITERATOR_GETKEY(iterator) && ":" &&
NSMAPS_MAP_ITERATOR_GETSTRING(iterator))
until NOT NSMAPS_MAP_ITERATOR_NEXT(iterator)
```

Voir aussi

[NSMAPS_MAP_STRING_NEW](#),
[NSMAPS_MAP_ITERATOR_GETVALUE](#)

[NSMAPS_MAP_ITERATOR_GETKEY](#),

Fonction [NSMAPS_MAP_GET_ITERATOR](#) (Librairie [NSMAPS](#))

Renvoie un itérateur depuis une Map. il servira à parcourir la Map.

Syntaxe	NSMAPS_MAP_GET_ITERATOR (<i>hMap</i>)
----------------	---

Paramètres	hMap	pointer	l	handle de la map
Valeur retournée	POINTER Itérateur que l'on pourra utiliser pour parcourir la map			

Remarque

L'itérateur ainsi récupéré pointe sur le premier élément de la liste. Cet itérateur devra être libéré par la suite.

Exemple :

```

local pointer myMap, pointer iterator

; boucle avec itérateur
trace ""
trace "--- boucle de parcours avec itérateur"
iterator = NSMAPS_MAP_GET_ITERATOR(mymap)
repeat
  trace "boucle Itérateur" && NSMAPS_MAP_ITERATOR_GETKEY(iterator)
until NOT NSMAPS_MAP_ITERATOR_NEXT(iterator)

; Libération de l'itérateur
NSMAPS_MAP_ITERATOR_DISPOSE (iterator)

```

Voir aussi

[NSMAPS_MAP_ITERATOR_GETKEY,](#)
[NSMAPS_MAP_ITERATOR_GETVALUE,](#)
[NSMAPS_MAP_ITERATOR_DISPOSE](#)

[NSMAPS_MAP_ITERATOR_GETSTRING,](#)
[NSMAPS_MAP_ITERATOR_NEXT,](#)

Fonction **NSMAPS_MAP_GET** (Librairie NSMAPS)

Retrouve un objet depuis une Map, à partir de sa clef.

Syntaxe	NSMAPS_MAP_GET (hMap, key\$)			
Paramètres	hMap	pointer	l	handle de la map
	key\$	cstring	l	clé de l'objet à retrouver depuis la map
Valeur retournée	POINTER Pointeur sur l'objet trouvé Si la valeur retournée vaut 0, l'objet n'a pas été trouvé.			

Exemple :

```

segment SEG_CLIENT
    nom$
    prenom$
    age%
endSegment

local pointer myMap
local SEG_CLIENT@ pCli
local key$

myMAP = NSMAPS_MAP_NEW

new @pCli
pCli.nom$ = "DUPOND"
pCli.prenom$ = "Jean"
pCli.age% = 54

key$ = pCli.nom$

NSMAPS_map_put (myMAP, key$, @pCli)

; relecture
@pCli = NSMAPS_MAP_GET (myMAP, "DUPOND")
if @pCli = 0
    trace "NOT Found"
else
    trace "Found:" && "pCli.nom$"
endif

```

Voir aussi

NSMAPS_MAP_PUT, NSMAPS_MAP_GET_STRING

Fonction NSMAPS_MAP_GET_STRING (Librairie NSMAPS)

Retrouve une DynStr depuis une Map, à partir de sa clef.

Syntaxe	NSMAPS_MAP_GET_STRING (<i>hMap, key\$</i>)			
Paramètres	hMap	pointer		handle de la map
	key\$	cstring		clé de la string à retrouver depuis la map
Valeur retournée	dynStr string dynamique retrouvée			

Remarque

Si la valeur retournée vaut "" (chaîne vide), l'objet n'a pas été trouvé.

Exemple :

```

local pointer myMap
local key$, dynStr DS

myMAP = NSMAPS_MAP_STRING_NEW

NSMAPS_map_put (myMAP, "KEY_1", "Dupond_1")
NSMAPS_map_put (myMAP, "KEY_2", "Dupond_1")

; relecture
DS = NSMAPS_MAP_GET_STRING (myMAP, "KEY_1")

```

Voir aussi

[NSMAPS_MAP_PUT_STRING](#), [NSMAPS_MAP_STRING_NEW](#)

Instruction **NSMAPS_MAP_DISPOSE** ([Librairie NSMAPS](#))

Supprime la map.

Syntaxe	NSMAPS_MAP_DISPOSE (<i>hMap</i>)		
Paramètres	hMap	pointer	handle de la map

Remarque

Si la map a été allouée par NSMAPS_MAP_NEW2, l'appel automatique à l'instruction de désallocation est effectuée pour tous les éléments de la map

Exemple :

```

segment SEG_CLIENT
  nom$
  prenom$
  age%
endSegment

local pointer myMap
local SEG_CLIENT@ pCli
local key$

myMAP = NSMAPS_MAP_NEW

new @pCli
pCli.nom$ = "DUPOND"
pCli.prenom$ = "Jean"
pCli.age% = 54

key$ = pCli.nom$

NSMAPS_map_put myMAP, key$, @pCli

```

NSMAPS_MAP_DISPOSE (hMap)

Voir aussi

NSMAPS_MAP_NEW, NSMAPS_MAP_NEW2

Fonction NSMAPS_MAP_ITERATOR_NEXT (Librairie NSMAPS)

Fait pointer l'itérateur sur l'élément suivant de la MAP.

Syntaxe	NSMAPS_MAP_ITERATOR_NEXT (<i>iterator</i>)		
Paramètres	iterator	pointer	I Itérateur récupéré pour parcourir la map
Valeur retournée	int(1) TRUE% si l'itérateur pointe sur un nouvel élément de la map, FALSE% si on est arrivé en fin de map		

Remarque

Quand la fin de la map est atteinte, l'itérateur ne peut plus être utilisé. Il devra être libéré.

Exemple :

```
local pointer myMap, pointer iterator
; boucle avec itérateur

trace "--- boucle de parcours avec itérateur"
iterator = NSMAPS_MAP_GET_ITERATOR(mymap)
repeat
    trace "boucle Itérateur" && NSMAPS_MAP_ITERATOR_GETKEY(iterator)
until NOT NSMAPS_MAP_ITERATOR_NEXT(iterator)

; Libération de l'itérateur
NSMAPS_MAP_ITERATOR_DISPOSE (iterator)
```

Voir aussi

NSMAPS_MAP_GET_ITERATOR, NSMAPS_MAP_ITERATOR_DISPOSE

Fonction NSMAPS_MAP_NEW (Librairie NSMAPS)

Création d'une map de pointeurs

Syntaxe	NSMAPS_MAP_NEW
Valeur retournée	POINTER handle de la map qui a été créée.

Voir aussi

NSMAPS_MAP_DISPOSE, **NSMAPS_MAP_NEW2**

Fonction NSMAPS_MAP_ITERATOR_GETKEY (Librairie NSMAPS)

Retrouve une clé depuis une Map, à partir de la position courante de l'itérateur.

Syntaxe	NSMAPS_MAP_ITERATOR_GETKEY (<i>iterator</i>)		
Paramètres	iterator	pointer	I itérateur parcourant la map
Valeur retournée	cstring clé de la position courante.		

Exemple :

```
local pointer myMap, pointer iterator
; boucle avec itérateur

trace "--- boucle de parcours avec itérateur"
iterator = NSMAPS_MAP_GET_ITERATOR(mymap)
repeat
    trace "boucle Iterateur" && NSMAPS_MAP_ITERATOR_GETKEY(iterator)
until NOT NSMAPS_MAP_ITERATOR_NEXT(iterator)

; Libération de l'itérateur
NSMAPS_MAP_ITERATOR_DISPOSE (iterator)
```

Voir aussi

NSMAPS_MAP_ITERATOR_GETSTRING, **NSMAPS_MAP_ITERATOR_GETVALUE**

Instruction NSMAPS_MAP_ITERATOR_DISPOSE (Librairie NSMAPS)

Supprime un itérateur.

Syntaxe	NSMAPS_MAP_ITERATOR_DISPOSE (<i>iterator</i>)			
Paramètres	iterator	pointer		itérateur à libérer

Exemple :

```
local pointer myMap, pointer iterator
; boucle avec itérateur
trace "--- boucle de parcours avec itérateur"
iterator = NSMAPS_MAP_GET_ITERATOR(mymap)
repeat
    trace "boucle Itérateur" && NSMAPS_MAP_ITERATOR_GETKEY(iterator)
until NOT NSMAPS_MAP_ITERATOR_NEXT(iterator)
; Libération de l'itérateur
NSMAPS_MAP_ITERATOR_DISPOSE (iterator)
```

Voir aussi

[NSMAPS_MAP_GET_ITERATOR](#)

Fonction NSMAPS_MAP_SIZE (Librairie NSMAPS)

Renvoie nombre d'éléments contenus dans une MAP

Syntaxe	NSMAPS_MAP_SIZE (<i>hMap</i>)			
Paramètres	hMap	pointer		handle de la map
Valeur retournée	int(4) nombre d'éléments contenus dans cette MAP			

Remarque

Cette fonction peut être utilisée avec les map de pointeurs et les map de string dynamiques.

Voir aussi

[NSMAPS_MAP_CONTAINS%](#)

Fonction NSMAPS_MAP_NEW2 (Librairie NSMAPS)

Création d'une map de pointeurs. le paramètre passé est l'adresse d'une INSTRUCTION utilisée par NSMAPS_MAP_REMOVE, NSMAPS_MAP_CLEAR et NSMAPS_MAP_DISPOSE pour libérer la mémoire allouée des DATA.

Syntaxe	NSMAPS_NEW2 (<i>inst_addr</i>)			
Paramètre	inst_addr	pointer	I	<i>adresse de l'instruction de désallocation</i>
Valeur retournée	POINTER handle de la map qui a été créée.			

Exemple :

```
instruction AUTO_DISPOSE pointer data
  dispose data
endInstruction ; AUTO_DISPOSE

instruction TEST_MAP_AUTO_DISPOSE

  local pointer myMap
  myMap = NSMAPS_MAP_NEW2(@AUTO_DISPOSE)
  ..
  ..
  NSMAP_MAP_DISPOSE (myMap)

endInstruction ; TEST_MAP_AUTO_DISPOSE
```

Voir aussi

NSMAPS_MAP_DISPOSE, NS_MAPS_NEW, NS_MAPS_CLEAR

Instruction **NSMAPS_MAP_PUT_STRING** (Librairie NSJSON)

Insère un objet (string ou dynStr) dans une map.

Syntaxe	NSMAPS_MAP_PUT_STRING (<i>hMap, key, DS</i>)			
Paramètres	hMap	pointer	I	handle de la map
	key	cstring	I	clé de la dynStr à insérer dans la map
	DS	dynStr	I	chaîne à insérer dans la map

Remarque

Une si une clef existe déjà, la chaîne précédente est remplacé par la nouvelle.

Exemple :

```
local key$, dynstr value$
local pointer myMap

myMAP = NSMAPS_MAP_STRING_NEW

key$ = "KEY_1"
value$ = "VALUE_1"
NSMAPS_map_put_string myMAP, key$, value$

key$ = "KEY_2"
value$ = "VALUE_2"
NSMAPS_map_put_string myMAP, key$, value$

; nombre d'elements
trace "nombre d'elements" && NSMAPS_map_size(myMap)

key$ = "KEY_1"
value$ = NSMAPS_map_get_string(myMAP, key$)
key$ = "KEY_2"
value$ = NSMAPS_map_get_string(myMAP, key$)

;Liberation de la map
NSMAPS_MAP_DISPOSE myMap
```

Voir aussi

[NSMAPS_MAP_STRING_NEW](#), [NSMAPS_MAP_GET_STRING](#),

Fonction [NSMAPS_MAP_STRING_NEW](#) ([Librairie NSMAPS](#))

Création d'une map de string dynamiques (dynStr)

Syntaxe	NSMAPS_MAP_STRING_NEW
Valeur retournée	POINTER handle de la map qui a été créée.

Voir aussi

[NSMAPS_MAP_DISPOSE](#), [NSMAPS_MAP_PUT_STRING](#), [NSMAPS_MAP_GET_STRING](#)

Instruction [NSMAPS_MAP_PUT](#) ([Librairie NSMAPS](#))

Insère un objet (pointeur) dans une map.

Syntaxe	NSMAPS_MAP_PUT (<i>hMap, key, data</i>)
----------------	--

Paramètres	hMap	pointer	I	handle de la map
	key	cstring	I	clé de l'objet à insérer dans la map
	data	pointer	I	pointeur sur l'objet à insérer dans la map

Remarque

Une si une clef existe déjà, le data précédent est remplacé par le nouveau. Si la map a été allouée par NSMAPS_MAP_NEW, l'ancien data peut ne pas être libéré. Si la map a été allouée par NSMAPS_MAP_NEW2, l'appel automatique à l'instruction de désallocation est effectuée pour l'ancien data

Exemple :

```
segment SEG_CLIENT
  nom$
  prenom$
  age%
endSegment

local pointer myMap
local SEG_CLIENT@ pCli
local key$

myMAP = NSMAPS_MAP_NEW

new @pCli
pCli.nom$ = "DUPOND"
pCli.prenom$ = "Jean"
pCli.age% = 54

key$ = pCli.nom$

NSMAPS_map_put myMAP, key$, @pCli
```

Voir aussi

[NSMAPS_MAP_GET](#), [NSMAPS_MAP_PUT_STRING](#)

Fonction NSMAPS_MAP_CONTAINS% ([Librairie NSMAPS](#))

Permet de savoir si une clé existe déjà dans une Map

Syntaxe	NSMAPS_MAP_CONTAINS% (<i>hMap, key\$</i>)			
Paramètres	hMap	pointer	I	handle de la map
	key\$	cstring	I	clé de l'objet à retrouver depuis la map

Valeur retournée	int(1) TRUE% si la clef existe déjà dans la MAP, FALSE% sinon
-----------------------------	---

Remarque

Cette fonction peut être utilisée avec les map de pointeurs et les map de string dynamiques.

Voir aussi

NSMAPS MAP PUT, *NSMAPS MAP GET*

Exemple NSMAPS (Librairie NSMAPS)

Voici un exemple d'utilisation de la librairie NSMAPS avec désallocation automatique de la mémoire allouée.

```
; client segment simple
segment SEG_CLIENT
  nom$
  prenom$
  age%
endSegment

;
; exemple de map de pointeurs avec allocation et désallocation en NCL
;
instruction TEST_SIMPLE_POINTER_MAP

  local pointer myMap
  local SEG_CLIENT@ pCli
  local key$
  local pointer iterator
  local index%

  traceClear

  myMAP = NSMAPS_MAP_NEW ; les pointeurs seront libérés avec la fonction de libération
  par défaut

  trace "--- Insertion dans la Map"

  ; boucle de remplissage avec put
  for index% = 7 to 12

    ; creation de pointeur(s)
    new @pCli
    pCli.nom$ = "DUPOND_" & string$(index%, "00")
    pCli.prenom$ = "Jean_" & index%
    pCli.age% = index%

    key$ = pCli.nom$

    NSMAPS_map_put myMAP, key$, @pCli
  endFor
```

```

; nombre d'elements
trace "nombre d'elements :" && NSMAPS_map_size(myMap)

; Boucle de parcours avec Get"
trace ""
trace "--- boucle de parcours avec Get"
for index% = 7 to 13
  key$ = "DUPOND_" & string$(index%, "00")
  @pCli = SEG_CLIENT(NSMAPS_map_get(myMAP, key$))
  if @pCli = 0
    trace "boucle Get" && index% && key$ && "NOT Found"
  else
    trace "boucle Get" && index% && key$ && ":" && pCli.nom$ & '/' & pCli.prenom$ &
    '/' & pCli.age%
  endif
endfor

; Test de NSMAPS_MAP_CONTAINS%
trace ""
trace "--- test de NSMAPS_MAP_CONTAINS%"
trace ("NSMAPS_MAP_CONTAINS% (DUPOND_10)=" & NSMAPS_MAP_CONTAINS%(mymap, "DUPOND_10"))
trace ("NSMAPS_MAP_CONTAINS% (DUPOND_14)=" & NSMAPS_MAP_CONTAINS%(mymap, "DUPOND_14"))

; Test de NSMAPS_MAP_REMOVE
trace ""
trace "--- test de NSMAPS_MAP_REMOVE"
if NSMAPS_MAP_CONTAINS%(mymap, "DUPOND_10")
  @pCli = SEG_CLIENT(NSMAPS_map_get(myMAP, "DUPOND_10"))
  dispose (@pCli)
  NSMAPS_MAP_REMOVE (mymap, "DUPOND_10")
  trace "nombre d'elements :" && NSMAPS_map_size(myMap)
endif

; boucle avec itérateur
trace ""
trace "--- boucle de parcours avec itérateur"
iterator = NSMAPS_MAP_GET_ITERATOR(mymap)
repeat
  @pCli = SEG_CLIENT(NSMAPS_MAP_ITERATOR_GETVALUE(iterator))
  trace "boucle Itérateur" && NSMAPS_MAP_ITERATOR_GETKEY(iterator) && ":" && pCli.nom$
  & '/' & pCli.prenom$ & '/' & pCli.age%
until NOT NSMAPS_MAP_ITERATOR_NEXT(iterator)

; Liberation de l'itérateur
NSMAPS_MAP_ITERATOR_DISPOSE iterator

; dispose des éléments
trace ""
trace "--- dispose pointers"
iterator = NSMAPS_MAP_GET_ITERATOR(mymap)
repeat
  @pCli = SEG_CLIENT(NSMAPS_MAP_ITERATOR_GETVALUE(iterator))
  trace "Manual dispose @pCli" && NSMAPS_MAP_ITERATOR_GETKEY(iterator) && ":" &&
  pCli.nom$ & '/' & pCli.prenom$ & '/' & pCli.age%
  dispose @pCli
until NOT NSMAPS_MAP_ITERATOR_NEXT(iterator)

; Liberation de l'itérateur
NSMAPS_MAP_ITERATOR_DISPOSE iterator

; Liberation de la map

```

```

NSMAPS_MAP_DISPOSE myMap

endInstruction ; TEST_SIMPLE_POINTER_MAP

; *****  desallocation automatique *****

; client segment avec adresse allouée

segment SEG_ADDR
  addr1$
  cp$
  ville$
endSegment ; SEG_ADDR

segment SEG_CLIENT_ADDR
  nom$
  prenom$
  age%
  SEG_ADDR@ addr
endSegment

;
; Desallocation automatiquement appelée par NSMAPS_MAP_DISPOSE
;
instruction AUTO_DISPOSE_ADDR_AND_PCLI pointer data

  local SEG_CLIENT_ADDR@ pCli

  @pCli = SEG_CLIENT_ADDR(data)
  trace "Automatic disapose @pCli & ADDR" && pCli.nom$ && pCli.addr.cp$

  dispose @pCli.addr
  dispose @pCli

endInstruction ; AUTO_DISPOSE_ADDR_AND_PCLI

;
; Exemple de map de pointeurs avec désallocation automatique
;
instruction TEST_MAP_AUTO_DISPOSE

  local pointer myMap
  local SEG_CLIENT_ADDR@ pCli
  local key$
  local pointer iterator
  local index%

  traceClear

  ; les pointeurs seront liberes avec la fonction de liberation indiquée
  myMAP = NSMAPS_MAP_NEW2(@AUTO_DISPOSE_ADDR_AND_PCLI)

  trace "--- Insertion dans la Map"

  ; boucle de remplissage avec put
  for index% = 7 to 12

    ; creation de pointeur(s)
    new @pCli
    pCli.nom$ = "DUPOND_" & string$(index%, "00")
    pCli.prenom$ = "Jean_" & index%
    pCli.age% = index%
    new @pCli.addr
    pCli.addr.addr1$ = index% & " rue des fleurs"

```

```

    pCli.addr.cp$ = 75000 + index%
    pCli.addr.ville$ = "Paris"

    key$ = pCli.nom$

    trace "insertion " && index% && key$ && ":" && pCli.nom$ & '/' & pCli.prenom$ & '/'
& pCli.age%

    NSMAPS_map_put myMAP, key$, @pCli
endFor

; nombre d'elements
trace "nombre d'elements :" && NSMAPS_map_size(myMap)

; Boucle de parcours avec Get"
trace ""
trace "--- boucle de parcours avec Get"
for index% = 7 to 13
    key$ = "DUPOND_" & string$(index%, "00")
    @pCli = SEG_CLIENT_ADDR(NSMAPS_map_get(myMAP, key$))
    if @pCli = 0
        trace "boucle Get" && index% && key$ && "NOT Found"
    else
        trace "boucle Get" && index% && key$ && ":" && pCli.nom$ & '/' & pCli.prenom$ &
'/' & pCli.age%
    endif
endFor

; Test de NSMAPS_MAP_CONTAINS%
trace ""
trace "--- test de NSMAPS_MAP_CONTAINS%"
trace ("NSMAPS_MAP_CONTAINS% (DUPOND_10)= " && NSMAPS_MAP_CONTAINS%(mymap, "DUPOND_10"))
trace ("NSMAPS_MAP_CONTAINS% (DUPOND_14)= " && NSMAPS_MAP_CONTAINS%(mymap, "DUPOND_14"))

; Test de NSMAPS_MAP_REMOVE
trace ""
trace "--- test de NSMAPS_MAP_REMOVE"
if NSMAPS_MAP_CONTAINS%(mymap, "DUPOND_10")
    NSMAPS_MAP_REMOVE (mymap, "DUPOND_10")
    trace "nombre d'elements :" && NSMAPS_map_size(myMap)
endif

; boucle avec itérateur
trace ""
trace "--- boucle de parcours avec itérateur"
iterator = NSMAPS_MAP_GET_ITERATOR(mymap)
repeat
    @pCli = SEG_CLIENT_ADDR(NSMAPS_MAP_ITERATOR_GETVALUE(iterator))
    trace "boucle itérateur" && NSMAPS_MAP_ITERATOR_GETKEY(iterator) && ":" && pCli.nom$
& '/' & pCli.prenom$ & '/' & pCli.age%
until NOT NSMAPS_MAP_ITERATOR_NEXT(iterator)

; Liberation de l'itérateur
NSMAPS_MAP_ITERATOR_DISPOSE iterator

; Liberation de la map
; Les pointeurs contenus sont aussi liberes car on a utilise NSMAPS_MAP_NEW2
trace ""
trace "--- map desallocation"
NSMAPS_MAP_DISPOSE myMap

endInstruction ; TEST_MAP_AUTO_DISPOSE

;
*****

```

```

*
;
; Exemple de map de DynStr. les désallocations sont automatiques
;
instruction TEST_DYNSTR_MAP

    local key$, dynstr value$
    local pointer myMap, pointer iterator
    local index%

    traceClear

    myMAP = NSMAPS_MAP_STRING_NEW

    ; boucle de remplissage avec put
    for index% = 7 to 12
        key$ = "KEY_" & string$(index%, "00")
        value$ = "VALUE_" & index%
        NSMAPS_map_put_string myMAP, key$, value$
    endFor

    ; nombre d'elements
    trace "nombre d'elements" && NSMAPS_map_size(myMap)

    ; boucle de parcours avec get
    trace ""
    trace "--- boucle de parcours avec Get"
    for index% = 7 to 13
        key$ = "KEY_" & string$(index%, "00")
        Trace ("boucle Get" && key$ && ":" && NSMAPS_map_get_string(myMAP, key$))
    endFor

    ; Test de NSMAPS_MAP_CONTAINS%
    trace ""
    trace "--- test de NSMAPS_MAP_CONTAINS%"
    trace ("NSMAPS_MAP_CONTAINS% (KEY_10)=" & NSMAPS_MAP_CONTAINS%(myMap, "KEY_10"))
    trace ("NSMAPS_MAP_CONTAINS% (KEY_14)=" & NSMAPS_MAP_CONTAINS%(myMap, "KEY_14"))

    ; Test de NSMAPS_MAP_REMOVE
    trace ""
    trace "--- test de NSMAPS_MAP_REMOVE"
    if NSMAPS_MAP_CONTAINS%(myMap, "KEY_10")
        NSMAPS_MAP_REMOVE (myMap, "KEY_10")
        trace "nombre d'elements :" && NSMAPS_map_size(myMap)
    endIf

    ; boucle avec itérateur
    trace ""
    trace "--- boucle de parcours avec itérateur"
    iterator = NSMAPS_MAP_GET_ITERATOR(myMap)
    repeat
        trace ("boucle Iterator" && NSMAPS_MAP_ITERATOR_GETKEY(iterator) && ":" &&
NSMAPS_MAP_ITERATOR_GETSTRING(iterator))
    until NOT NSMAPS_MAP_ITERATOR_NEXT(iterator)

    ; Liberation de l'itérateur
    NSMAPS_MAP_ITERATOR_DISPOSE iterator

    ;Liberation de la map
    NSMAPS_MAP_DISPOSE myMap

endInstruction ; TEST_DYNSTR_MAP

```



```
;
; TRACE
;
instruction traceClear
  delete from wMAPS(mainwindow%).LB_RESULT
endInstruction ; trace

instruction trace s$
  insert at end s$ to wMAPS(mainwindow%).LB_RESULT
endInstruction ; trace
```

Voir aussi : [Librairie NSMAPS](#)

LIBRAIRIE NSDYNCOLLEC

Cette librairie permet de gérer une collection dynamique de pointeurs

Cette librairie permet de gérer facilement une collection de pointeurs, et offre de nombreuses fonctionnalités.

- Ajustement automatique de la taille du tableau.
- Insertion à n'importe quelle endroit du tableau
- récupération des élément par les N° d'indexation ou leur valeur.
- Tri du tableau avec fonction de comparaison personnalisée.
- Libération automatique, sur flag des éléments du tableau
- Facilité d'interrogation du tableau.

Elle peut être utilisée par, par exemple, pour mémoriser une liste de segment sans avoir à se préoccuper de leur nombre.

Et d'effectuer toutes les opération d'insertion, de tri, de suppression facilement.

Elle permet également de fusionner plusieurs tableaux

Référence de la librairie NSDYNCOLLEC

Voici la liste des instructions et fonctions de la librairie NSDYNCOLLEC regroupées par catégories.

Création de nouveaux Tableaux

Crée un nouveau tableau dynamique

function DA create

Interrogation des propriétés du tableau

Retourne le nombre d'éléments dans ce tableau.

function DA size%

Retourne la capacité actuelle de ce tableau.

function DA capacity%

Vérifie si ce tableau ne contient aucun élément.

function DA isEmpty%

Vidage et suppression du tableau

Supprime tous les éléments du tableau dynamique

function DA_clear%

Supprime tous les éléments d'un tableau dynamique et détruit le tableau

function DA_delete%

Ajout et remplacement d'éléments dans le tableau

Ajoute l'élément spécifié à la fin de ce tableau dynamique

function DA_append%

Insère l'élément à l'index spécifié.

function DA_insertAt%

Remplace un du tableau par un nouvel élément

function DA_setAt

Interrogation des éléments du tableau

Retourne le premier élément de ce tableau.

function DA_getFirst

Retourne le dernier élément de ce tableau.

function DA_getLast

Retourne l'élément à l'index spécifié de ce tableau.

function DA_getAt

renvoie l'index de la première occurrence de l'élément.

function DA_indexOf%

renvoie l'index de la dernière occurrence de l'élément.

function DA_lastIndexOf%

renvoie vrai si le tableau dynamique contient l'élément spécifié, faux sinon.

function DA_contains%

Suppression d'éléments dans le tableau

Supprime le composant à l'index spécifié dans ce tableau.

function DA_removeAt%

Supprime la première occurrence de l'élément spécifié dans ce tableau.

function DA_remove%

Tri du tableau

trie les éléments du tableau dynamique.

instruction DA_sort

Gestion fine de l'agrandissement du tableau

Crée un nouveau tableau dynamique avec une capacité et un pourcentage d'incrément spécifiés.

function DA createWithCapacity

réserve de l'espace pour de nouveaux éléments.

function DA reserve%

Gestion multi tableaux

insère un tableau statique spécifié dans un autre tableau dynamique à l'index spécifié.

function DA insertArrayAt

insère un tableau dynamique spécifié dans un autre tableau dynamique à l'index spécifié.

function DA insertDynArrayAt

Il est important de noter que ces fonctions utilisent des pointeurs pour accéder aux tableaux dynamiques et aux éléments,

il est donc important de gérer correctement les pointeurs lors de l'utilisation de ces fonctions.

.

Exemple DYNCOLLEC (Librairie NSDYNCOLLEC)

Voici un exemple d'utilisation de la librairie NSMAPS avec désallocation automatique de la mémoire allouée.

```
;
;
;
; Exemple de collection dynamique de pointeurs avec désallocation personnalisée
automatique
;
```

```
;
; segment SEG_PERSON
;
segment SEG_PERSON
  name$
  age%
```

```

    dynstr firstName$
endSegment ; SEG_PERSON

;
;Instruction de désallocation de la mémoire.
;
Instruction SDA_MY_DISPOSE pointer myElem
    dispose (myElem)
EndInstruction ; SDA_MY_DISPOSE

```

```

local pointer my_da
local SEG_PERSON@ pDat
local SEG_PERSON@ pDat2
local u%

my_da = DA_create(@SDA_MY_DISPOSE)

; remplissage
for u% = 0 to 9
    new @pDat
    Dat.name$ = "DUPONT_" & u%
    pDat.age% = u% * 10
    pDat.firstName$ = "Jean_" & u%
    if not DA_append%(my_da, @pDat)
        message "Error appel DA_append% ", u%
    endIf
endFor

; liste des éléments
insert at end "Original list" to LB_RESULT
for u% = 0 to DA_size%(my_da) - 1
    @pDat = SEG_PERSON(DA_getAt(my_da, u%))
    insert at end u% && pDat.name$ && pDat.firstName$ && pDat.age% to LB_RESULT
endFor

; remplacement d'un élément
new @pDat2
pDat2.name$ = "MARTIN_MARTIN"
pDat2.age% = 100
pDat2.firstName$ = "jacques"
@pDat = SEG_PERSON(DA_setAt(my_da, @pDat2, 3))
insert at end "Old Elem =" && pDat.name$ && pDat.firstName$ && pDat.age% to LB_RESULT
dispose @pDat

; insertion d'un élément
new @pDat
pDat.name$ = "Hochon"
pDat.age% = 101
pDat.firstName$ = "paul"
if DA_insertAt%(my_da, @pDat, 2) <> true%
    message "Error appel DA_insertAt% ", 2
endIf

; suppression de l'élément 5
DA_removeAt% (my_da, 5, False%)
; liste des éléments
insert at end "final list" to LB_RESULT
for u% = 0 to DA_size%(my_da) - 1

```

```
@pDat = SEG_PERSON(DA_getAt(my_da, u%))
insert at end u% && pDat.name$ && pDat.firstName$ && pDat.age% to LB_RESULT
endFor

; Suppression de la lise avec appel automatique à l'instruction de désallocation mémoire.
DA_delete% (my_da, true%)
```

Voir aussi : [Librairie NSDYNCOLLEC](#)

Fonction Da_append% ([Librairie NSDYNCOLLEC](#))

Ajoute un objet à la fin du tableau dynamique.

Syntaxe	Function Da_append% (pointer dynArray, pointer pElem)			
Paramètres	dynArray	pointer	I	handle du tableau dynamique
	pElem	pointer	I	pointeur de l'objet à ajouter au tableau
Valeur retournée	INT(4)	True% si l'ajout a réussi, False% sinon		

Remarques

Si on veut ajouter un élément à un tableau qui est plein, celui-ci s'agrandit automatiquement.

Le seul cas où cet ajout peut échouer, c'est quand le système n'a plus du tout de mémoire disponible, et qu'il ne peut pas en allouer. Ce qui n'arrive jamais.

Voir aussi

[NSMAPS_SIZE, DA_INSERTAT%](#)

Fonction DA_capacity% ([Librairie NSDYNCOLLEC](#))

Retourne la taille actuelle maximale du tableau

Syntaxe	Function Da_size% (pointer dynArray)			
Paramètres	dynArray	pointer	I	handle du tableau dynamique
Valeur retournée	INT(4)	Taille maximale actuelle du tableau		

Remarque

Cette fonction renvoie la taille actuelle maximale du tableau, indépendamment du nombre d'éléments contenus..

Voir aussi

DA_SIZE%, DA_CREATEWITHCAPACITY

Fonction DA_clear% (Librairie NSDYNCOLLEC)

Supprime tous les éléments du le tableau

Syntaxe	Function DA_clear (pointer dynArray, bDisposeElem%)			
Paramètres	dynArray	pointer	I	handle du tableau dynamique
	bDisposeElem%	INT(4)	I	TRUE% si on veut que les pointeurs soient désalloués.
Valeur retournée	INT(4)	True% ou False% si l'index est en dehors des limites du tableau dynamique.		

Remarques

Les désallocations peuvent appeler automatiquement l'instruction (CallBack) de désallocation utilisateur précisée dans la commande DA_CREATE

Voir aussi

DA_DELETE

Fonction DA_CONTAINS% (Librairie NSDYNCOLLEC)

Permet de savoir si une un pointeur est déjà présent dans une collection.

Syntaxe	Function DA_contains% (pointer dynArray, pointer pElem)			
Paramètres	dynArray	pointer	I	handle du tableau dynamique
	pElem	pointer	I	pointeur de l'objet à retrouver depuis le tableau
Valeur retournée	INT(4)	True% si le pointeur est déjà dans le tableau., False% sinon		

Remarque

Cette fonction peut être utilisée avec un tableau de pointeurs ou un tableau d'entier.

Voir aussi

DA_INDEXOF

Fonction DA_CREATE (Librairie NSDYNCOLLEC)

Création d'un nouveau tableau dynamique.

Syntaxe	Function DA_create (pointer pDestruct)		
Paramètres	pDestruct	pointer	pointeur vers la fonction de destruction de l'objet. peut être NULL
Valeur retournée	pointer	pointeur qui permet d'accéder au nouveau tableau créé	

Remarque

Toutes les autres fonction et instruction DA_* utiliseront ce pointeur de référence.

- Si le pointeur vers le destructeur est précisé, il doit pointer vers un instruction dont le prototype est le suivant :

```
; INSTRUCTION MY_DESTROY_ELEM pointer pElem
```

cette instruction devra libérer la ou les mémoires alloués par le pointeur.

- Si le paramètre passé est 0, (ou #null depuis la version 12) un dispose standard NCL sera effectué sur le pointeur si nécessaire.

exemple

```
; tableau dynamique avec dispose standard.
local pointer my_da

my_da = DA_create(#null)
...
...
DA_delete (my_da, true%) ; True% entraine la libération de l mémoire.

; Tableau dynamique avec callback de libération mémoire
local pointer my_da

Instruction MY_DESTROY_ELEM pointer pElem
  dispose pElem
endInstruction

my_da = DA_create(@MY_DESTROY_ELEM)
...
```

```

...
Da_delete (my_da, true%) ; True% entraine la libération de tous les pointeurs alloués.
;
; dans ce cas l'instruction de libération (MY_DESTROY_ELEM) sera appelée à chaque
libération de pointeur.
;

```

Voir aussi

[DA_REMOVE](#), [DA_CLEAR](#), [DA_CREATEWITHCAPACITY](#),

Fonction **DA_CreateWithCapacity** (Librairie NSDYNCOLLEC)

Création d'un nouveau tableau dynamique. en précisant les capacités d'agrandissement.

Syntaxe	Function DA_CreateWithCapacity (pointer pDestruct, initCapacity%, incrPercent%)			
Paramètres	pDestruct	pointer	I	pointeur vers la fonction de destruction de l'objet. peut être NULL
	initCapacity%	int	I	Capacité initiale du tableau
	incrPercent%	int	I	Incrément d'agrandissement en pourcentage.
Valeur retournée	pointer	pointeur qui permet d'accéder au nouveau tableau créé		

Cette fonction a le même usage que DA_CREATE, mais on lui précise la capacité initiale du tableau et la façon dont il s'agrandit. Elle

Remarque

La fonction DA_CREATE utilise la fonction en lui passant comme paramètres . initCapacity:100 et enchrPercent:30

Ce qui signifie que les tableaux dynamiques standard commencent avec une capacité initiale de 100 pointeurs, et qu'ils s'agrandissent de 30% quand c'est nécessaire

Voir aussi

[DA_CREATE](#), [DA_DELETE%](#)

Fonction **DA_delete%** (Librairie NSDYNCOLLEC)

Supprime le tableau dynamique et facultativement tous les éléments du le tableau

Syntaxe	Function DA_delete% (pointer @dynArray, bDisposeElem%)			
Paramètres	dynArray	pointer	I	handle du tableau dynamique
	bDisposeElem%	INT(4)	I	TRUE% si on veut que les pointeurs soient désalloués.

Valeur retournée	INT(4)	True% ou False% si l'index est en dehors des limites du tableau dynamique.
-------------------------	--------	--

Remarques

Les désallocations peuvent appeler automatiquement l'instruction (CallBack) de désallocation utilisateur précisée dans la commande DA_CREATE

Le pointeur de la collection est remis à 0 une fois la suppression effectuée.

Voir aussi

DA_CREATE, DA_CLEAR

Fonction Da_getAt (Librairie NSDYNCOLLEC)

Retourne l'élément à l'index spécifié

Syntaxe	Function Da_getAt (pointer dynArray, index%)			
Paramètres	dynArray	pointer	I	handle du tableau dynamique
	index%	int(4)	I	Index de l'élément
Valeur retournée	pointer	élément à la position indiquée du tableau 0 si l'index est en dehors des capacités du tableau		

Voir aussi

DA_GETLAST, DA_GETFIRST, DA_INSERTAT%

Fonction Da_getFirst (Librairie NSDYNCOLLEC)

Retourne le premier élément du tableau dynamique

Syntaxe	Function Da_getFirst (pointer dynArray)			
Paramètres	dynArray	pointer	I	handle du tableau dynamique
Valeur retournée	pointer	Le premier élément du tableau si celui-ci n'est pas vide 0 si le tableau est vide		

Voir aussi

DA_GETLAST, DA_GETAT

Fonction Da_getLast (Librairie NSDYNCOLLEC)

Retourne le dernier élément du tableau dynamique

Syntaxe	Function Da_getLast (pointer dynArray)			
----------------	--	--	--	--

Paramètres	dynArray	pointer		handle du tableau dynamique
Valeur retournée	pointer	Le dernier élément du tableau si celui-ci n'est pas vide 0 si le tableau est vide		

Voir aussi

DA_GETFIRST, DA_GETAT

Fonction Da_indexOf% (Librairie NSDYNCOLLEC)

Retourne l'index de l'objet passé en paramètre.

Syntaxe	Function Da_indexOf% (pointer dynArray, pointer pElem)			
Paramètres	dynArray	pointer		handle du tableau dynamique
	pElem	pointer		pointeur de l'objet à retrouver depuis le tableau
Valeur retournée	INT(4)	si trouvé, index dans le tableau constante DYN_ARRAY_NOT_FOUND si non trouvé,		

Voir aussi

DA_LASTINDEXOF%, DA_CONTAINS%

Fonction DA_insertArrayAt% (Librairie NSDYNCOLLEC)

Insère un tableau d'entier ou de pointeur dans une collection dynamique.

Syntaxe	Function DA_insertArrayAt% (Pointer pTrgDynArray, index%, pointer pSrcArray, first%, nbElem%)			
Paramètres	pTrgDynArray	pointer		handle du tableau dynamique dans lequel on veut insérer le tableau de pointeurs.
	index%	int(4)		Emplacement dans le tableau dynamique ou on veut insérer.
	pSrcArray	pointer		Adresse du tableau de pointeur que l'on veut insérer
	first%	int(4)		indices du 1er élément du tableau source que l'on veut insérer

	nbElem%	int(4)		Nombre d'éléments du tableau source que l'on veut insérer.
Valeur retournée	INT(4)	True% ou False% si l'allocation mémoire n'a pu se faire.		

Remarques

Lors de la suppression de la collection dynamique par la commande DA_CLEAR ou DA_DELETE, si le paramètre bDisposeElem % est à True%, la désallocation s'effectue également sur les pointeur insérés depuis le tableau d'entrée de pointeurs.

Si précisée comme paramètre dans l'instruction DA_CREATE, l'instruction de désallocation utilisateur est appelée pour tous les éléments du nouveau tableau.

Voir aussi

DA_INSERTDYNARRAYAT%

Fonction DA_InsertAt% (Librairie NSDYNCOLLEC)

Insère un nouvel élément à une index précis dans le tableau

Syntaxe	Function DA_insertAT% (pointer dynArray, pointer pElem, index%)			
Paramètres	dynArray	pointer		handle du tableau dynamique
	pElem	pointer		pointeur de l'objet à retrouver depuis le tableau
	index%	INT(4)		emplacement dans le tableau ou insérer l'élément
Valeur retournée	INT(4)	True% si le pointeur est bien inséré dans le tableau,. False% sinon		

Remarques

Le tableau grandit et est décalé à partir de l'élément inséré.

Cette fonction peut renvoyer FALSE% si l'index est en dehors des limites du tableau dynamique.

Voir aussi

DA_APPEND

Fonction DA_insertDynArrayAt% (Librairie NSDYNCOLLEC)

Insère un tableau dynamique dans un autre tableau dynamique.

Syntaxe	function DA_insertDynArrayAt%(Pointer pTrgDynArray, index%, Pointer pSrcDynArray, first%, nbElem%)			
Paramètres	pTrgDynArray	pointer	I	handle du tableau dynamique dans lequel on veut insérer le tableau de pointeurs.
	index%	int(4)	I	Emplacement dans le tableau dynamique ou on veut insérer.
	pSrcDynArray	pointer	I	handle du tableau dynamique que l'on veut insérer
	first%	int(4)	I	indice du 1er élément du tableau source que l'on veut insérer
	nbElem%	int(4)	I	Nombre d'éléments du tableau source que l'on veut insérer.
Valeur retournée	INT(4)	True% ou False% si l'allocation mémoire n'a pu se faire. ou si les index sont en dehors des bornes.		

Remarques

Lors de la suppression de la collection dynamique par la commande DA_CLEAR ou DA_DELETE, si le paramètre bDisposeElem % est à True%, la désallocation s'effectue également sur les pointeur insérés depuis le tableau d'entrée de pointeurs.

Si précisée comme paramètre dans l'instruction DA_CREATE, l'instruction de désallocation utilisateur est appelée pour tous les éléments du nouveau tableau.

Voir aussi

DA_INSERTARRAYAT%

Fonction DA_isEmpty% (Librairie NSDYNCOLLEC)

Permet de savoir si une collection est vide d'éléments

Syntaxe	Function DA_isEmpty% (pointer dynArray)			
Paramètres	dynArray	pointer	I	handle du tableau dynamique

Valeur retournée	INT(4)	True% si la collection de contient aucun élément. False% sinon
-------------------------	--------	---

Voir aussi

DA_INDEXOF%

Fonction DA_LastIndexOf% (Librairie NSDYNCOLLEC)

Retourne le dernier index de l'objet passé en paramètre.

Syntaxe	Function DA_LastIndexOf% (pointer dynArray, pointer pElem)			
Paramètres	dynArray	pointer	I	handle du tableau dynamique
	pElem	pointer	I	pointeur de l'objet à retrouver depuis le tableau
Valeur retournée	INT(4)	si trouvé, dernier index de l'objet dans le tableau constante DYN_ARRAY_NOT_FOUND si non trouvé,		

Remarque

S'il n'y a qu'une seule occurrence de l'objet dans le tableau, les fonctions DA_indexOf% et DA_LastIndexOf% renvoient le même index.

Voir aussi

DA_INDEXOF

Fonction DA_remove% (Librairie NSDYNCOLLEC)

Supprime un élément dont on connaît la valeur, mais pas l'index, dans le tableau

Syntaxe	Function DA_remove% (pointer dynArray, pointer pElem, bDisposeElem%)			
Paramètres	dynArray	pointer	I	handle du tableau dynamique
	pElem	pointer	I	Elément que l'on veut supprimer du tableau dynamique.
	bDisposeElem%	INT(4)	I	TRUE% si on veut que ce pointeur soit désalloué.
Valeur retournée	INT(4)	True% ou False% si l'index est en dehors des limites du tableau dynamique, ou si l'élément n'est pas trouvé dans le tableau		

Remarques

La désallocation peut appeler automatiquement l'instruction (CallBack) de désallocation utilisateur si elle a été précisée via la commande DA_CREATE

Cette fonction peut être utilisée pour supprimer un élément dont on connaît la valeur mais pas la position dans le tableau.

Voir aussi

DA_REMOVEAT%, DA_INDEXOF%

Fonction DA_removeAt% (Librairie NSDYNCOLLEC)

Supprime un élément à un index précis dans le tableau

Syntaxe	Function DA_removeAt% (pointer dynArray, index%, bDisposeElem%)			
Paramètres	dynArray	pointer	I	handle du tableau dynamique
	index%	INT(4)	I	emplacement dans le tableau ou supprimer l'élément
	bDisposeElem%	INT(4)	I	TRUE% si on veut que ce pointeur soit désalloué.
Valeur retournée	INT(4)	True% ou False% si l'index est en dehors des limites du tableau dynamique.		

Remarques

La désallocation peut appeler automatiquement l'instruction (CallBack) de désallocation utilisateur précisée dans la commande DA_CREATE

Voir aussi

DA_INSERTAT%, DA_REMOVE%

Fonction Da_reserve% (Librairie NSDYNCOLLEC)

Alloue de la place dans le tableau pour stocker un certain nombre de nouvelles valeurs..

Syntaxe	Function Da_reserve% (pointer dynArray, nbNew%)			
Paramètres	dynArray	pointer	I	handle du tableau dynamique

	int(4)	nbNew%	I	Nombre d'éléments à insérer dans le tableau
Valeur retournée	INT(4)	True%, si l'allocation a pu être faite, False% sinon		

Remarque

Cette fonction n'a d'utilité que si on veut allouer en une seule fois un très grand tableau.

Voir aussi

[DA_CREATE](#), [DA_CREATEWITHCAPACITY](#), [DA_CAPACITY%](#)

Fonction DA_setAt (Librairie NSDYNCOLLEC)

Remplace un élément par un autre élément à un index précis dans le tableau

Syntaxe	Function DA_setAt% (pointer dynArray, pointer pElem, index%)			
Paramètres	dynArray	pointer	I	handle du tableau dynamique
	pElem	pointer	I	pointeur de l'objet à retrouver depuis le tableau
	index%	INT(4)	I	emplacement dans le tableau ou remplacer l'élément
Valeur retournée	pointer	pointeur vers l'ancien l'élément remplacé par le nouveau.		

Remarques

Il peut être utile de désallouer le pointeur retourné..

Cette fonction peut renvoyer 0 si l'index est en dehors des limites du tableau dynamique.

Voir aussi

[DA_INSERTAT](#), [DA_REMOVEAT%](#)

Fonction DA_size% (Librairie NSDYNCOLLEC)

Retourne le nombre d'éléments présents dans le tableau

Syntaxe	Function Da_size% (pointer dynArray)			
Paramètres	dynArray	pointer	I	handle du tableau dynamique

Valeur retournée	INT(4)	nombre d'élément contenus dans le tableau
-------------------------	--------	---

Voir aussi

DA_CAPACITY

instruction **DA_sort** (Librairie NSDYNCOLLEC)

Trie un tableau en utilisant une instruction personnalisée de comparaison d'éléments.

Syntaxe	Instruction DA_sort% (pointer dynArray, ascending%, pointer pMyCompare)		
Paramètres	dynArray	pointer	handle du tableau dynamique
	ascending	int(4)	true% si on veut un tri croissant, false% sinon
	pMyCompare%	pointer	Adresse de la fonction de comparaison

Remarques

Le prototype de la fonction de comparaison doit être de la forme suivante.

```

;*****
;*****
; Sort managment
;
; Function my_compare% (MY_SEGMENT @item1, MY_SEGMENT @iTem2)
;
; comparison function prototype (if array of MY_SEGMENT) Must return :
;
; - a negative value if element item1% is less than element item2%,
; - 0 if they are identical
; - and a strictly positive value if element item1% is greater than element item2%
;
;*****
;*****

```

La fonction de comparaison doit renvoyer :

- une valeur négative si l'élément item1% est inférieur à l'élément item2%,
- 0 s'ils sont identiques
- et une valeur strictement positive si l'élément item1% est supérieur à l'élément item2%.

exemple

Déclaration d'un segment et fonction de tri par age.

```
;
; SEG_PERSON
;
segment SEG_PERSON
  name$
  age%
  dynstr firstName$
endSegment ; SEG_PERSON

;
; segment compariron by age
;
function SDA_my_compare_Pers_by_age%(SEG_PERSON@ s1, SEG_PERSON@ s2)
  return s1.age% - s2.age%
endFunction ; SDA_my_compare_Pers%
```

Utilisation du tableau et tri avec la fonction de comparaison.

```
local pointer my_da, u%
local SEG_PERSON@ pDat

my_da = DA_create(0)
for u% = 0 to 9
  new @pDat
  pDat.name$ = "DUPONT_" & u%
  pDat.age% = random%(10000)
  pDat.firstName$ = "Jean_" & u%
  if not DA_append%(my_da, @pDat)
    message "Error appel DA_append% ", u%
  endif
endfor

insert at end "----- initial values -----" to LB_RESULT
for u% = 0 to DA_size%(my_da) - 1
  @pDat = SEG_PERSON(DA_getAt(my_da, u%))
  insert at end u% && pDat.name$ && pDat.firstName$ && pDat.age% to LB_RESULT
endfor

Da_sort (my_da, true%, @SDA_my_compare_Pers_by_age%)
insert at end "----- after Sort -----" to LB_RESULT
for u% = 0 to DA_size%(my_da) - 1
  @pDat = SEG_PERSON(DA_getAt(my_da, u%))
  insert at end u% && pDat.name$ && pDat.firstName$ && pDat.age% to LB_RESULT
endfor

DA_delete (my_da, true%)
```


LIBRAIRIE DE GESTION DE LOG NCLLOGR

Cette librairie permet de gérer la génération de fichier LOG, en utilisant toute la puissance de LOG4J/LOG4CXX

Vous pouvez utiliser dans votre code applicatif la nouvelle librairie nclLogr.NCL (remplaçante de NSLOGR.NCL) pour ajouter vos propres logs.

La fonctionnalité NS_TRACE est également envoyée par défaut vers ce nouveau logger. Elle devient obsolète. Nous vous recommandons l'utilisation de la librairies nclLogr.NCL qui permet :

- L'utilisation d'un des 6 niveaux de log : TRACE, DEBUG, INFO, WARN, ERROR, FATAL
- L'utilisation de plusieurs instances de logger qui permettent une activation différenciée des niveaux de log visibles et un filtrage efficace.

Ce mécanisme de log est compatible avec la production et fonctionne en mode serveur.

Il gère les accès concurrents, ce qui permet à plusieurs applications ou plusieurs utilisateurs d'alimenter en même temps le même fichier.

Le runtime NS-DK utilise log4cxx pour loguer les erreurs les plus importantes avec leur pile (uniquement sous Windows pour la pile).

Par défaut, ce mécanisme est actif si l'application trouve le fichier NSLOGR.INI dans l'un des répertoires suivant :

1. Répertoire courant. (.)
2. Répertoire du programme
3. (NS-CFG) pour Windows or (NS_CFG) sinon
4. (NS-INI) pour Windows or (NS_INI) sinon, en enlevant le nom de fichier de terminaison si présent.
5. Windows seulement: (LOCALAPPDATA)exeName
6. Windows seulement: (APPDATA)exeName
7. Windows seulement: (ALLUSERSPROFILE)exeName

8. (Répertoire du programme).\ini
9. Tous les répertoires contenus dans la variable d'environnement PATH)

Il est possible de désactiver le nouveau mécanisme en positionnant

```
[Miscellaneous]
initLogger=False
```

dans le fichier nslib.ini : votre application log comme auparavant.

Référence de la librairie nclLOGR

Référence de la librairie nclLOGR

Voici la liste des instructions et fonctions de la librairie nclLOGR regroupées par catégories.

Constantes de niveau de log

Constantes LL *

Interrogation du fichier NSLOG.INI

Fonction NSIsLogLevelEnabled

Trace dans le fichier log

Instruction nsLogTrace

Instruction nsLogDebug

Instruction nsLogInfo

Instruction nsLogWarn

Instruction nsLogError

Instruction nsLogFatal

Exemple d'utilisation

Exemple nclLOGR

Instruction nsLogTrace (Librairie NCLLOGR)

Insère un texte au niveau **TRACE** dans un traceur particulier pour écrire dans le fichier de log.

Syntaxe	NSLogTrace (<i>logger\$, msg\$</i>)
----------------	---------------------------------------

Paramètres	logger\$	cstring	I	nom du traceur où écrire
	msg\$	dynStr	I	texte à écrire

Remarque

Si l'écriture du log est effectuée pour un niveau qui a été désactivé, aucune action de sera effectué.

Exemple :

Voir *l'exemple fourni*

Voir aussi

nsLogTrace, nsLogDebug, nsLogInfo, nsLogWarn, nsLogError, nsLogFatal

Instruction nsLogDebug (Librairie NCLLOGR)

Insère un texte au niveau **DEBUG** dans un traceur particulier pour écrire dans le fichier de log.

Syntaxe	NSLogTrace (<i>logger\$, msg\$</i>)			
Paramètres	logger\$	cstring	I	nom du traceur où écrire
	msg\$	dynStr	I	texte à écrire

Remarque

Si l'écriture du log est effectuée pour un niveau qui a été désactivé, aucune action de sera effectué.

Exemple :

Voir *l'exemple fourni*

Voir aussi

nsLogTrace, nsLogDebug, nsLogInfo, nsLogWarn, nsLogError, nsLogFatal

Instruction nsLogInfo (Librairie NCLLOGR)

Insère un texte au niveau **INFO** dans un traceur particulier pour écrire dans le fichier de log.

Syntaxe	NSLogInfo (<i>logger\$, msg\$</i>)			
Paramètres	logger\$	cstring	I	nom du traceur où écrire
	msg\$	dynStr	I	texte à écrire

Remarque

Si l'écriture du log est effectuée pour un niveau qui a été désactivé, aucune action de sera effectué.

Exemple :

Voir *[l'exemple fourni](#)*

Voir aussi

[nsLogTrace](#), *[nsLogDebug](#)*, *[nsLogInfo](#)*, *[nsLogWarn](#)*, *[nsLogError](#)*, *[nsLogFatal](#)*

Instruction NSLogInfo (Librairie NCLLOGR)

Insère un texte au niveau **INFO** dans un traceur particulier pour écrire dans le fichier de log.

Syntaxe	NSLogInfo (<i>logger\$, msg\$</i>)			
Paramètres	logger\$	cstring	I	nom du traceur où écrire
	msg\$	dynStr	I	texte à écrire

Remarque

Si l'écriture du log est effectuée pour un niveau qui a été désactivé, aucune action de sera effectué.

Exemple :

Voir *[l'exemple fourni](#)*

Voir aussi

[nsLogTrace](#), *[nsLogDebug](#)*, *[nsLogInfo](#)*, *[nsLogWarn](#)*, *[nsLogError](#)*, *[nsLogFatal](#)*

Instruction NSLogWarn (Librairie NCLLOGR)

Insère un texte au niveau **WARNING** dans un traceur particulier pour écrire dans le fichier de log.

Syntaxe	NSLogWarn (<i>logger\$, msg\$</i>)			
Paramètres	logger\$	cstring		nom du traceur où écrire
	msg\$	dynStr		texte à écrire

Remarque

Si l'écriture du log est effectuée pour un niveau qui a été désactivé, aucune action de sera effectué.

Exemple :

Voir *l'exemple fourni*

Voir aussi

nsLogTrace, nsLogDebug, nsLogInfo, nsLogWarn, nsLogError, nsLogFatal

Instruction NSLogError (Librairie NCLLOGR)

Insère un texte au niveau **ERROR** dans un traceur particulier pour écrire dans le fichier de log.

Syntaxe	NSLogError (<i>logger\$, msg\$</i>)			
Paramètres	logger\$	cstring		nom du traceur où écrire
	msg\$	dynStr		texte à écrire

Remarque

Si l'écriture du log est effectuée pour un niveau qui a été désactivé, aucune action de sera effectué.

Exemple :

Voir *l'exemple fourni*

Voir aussi

nsLogTrace, nsLogDebug, nsLogInfo, nsLogWarn, nsLogError, nsLogFatal

Instruction NSLogFatal (Librairie NCLLOGR)

Insère un texte au niveau **FATAL** dans un traceur particulier pour écrire dans le fichier de log.

Syntaxe	NSLogFatal (<i>logger\$, msg\$</i>)			
Paramètres	logger\$	cstring		nom du traceur où écrire
	msg\$	dynStr		texte à écrire

Remarque

Si l'écriture du log est effectuée pour un niveau qui a été désactivé, aucune action ne sera effectuée.

Exemple :

Voir *l'exemple fourni*

Voir aussi

nsLogTrace, nsLogDebug, nsLogInfo, nsLogWarn, nsLogError, nsLogFatal

Fonction NSIsLogLevelEnabled% (Librairie NnlLogr)

Renvoie vrai si le niveau de log choisi est activé pour un traceur précis.

Syntaxe	Function NSIsLogLevelEnabled% (<i>level%, logger\$</i>)			
Paramètres	level%	INT(1)		Niveau à tester (constante LL_*)
	logger\$	cstring		Nom du traceur à tester
Valeur retournée	INT(1)	True% ou False% si le traceur est activé pour le niveau choisi.		

Voir aussi

*constantes LL_**

Exemple

Si dans le fichier NSLOGR.INI, on a la définition suivante,

```
log4j.logger.fr.natsystem.myAppli=INFO, myAppliLog
```

le code NCL suivant

```
local u%, ret%, Logger$
Logger$ = "fr.natsystem.myAppli"
for u% = LL_TRACE% to LL_FATAL%
    ret% = NSIsLogLevelEnabled% (u%,Logger$)
    insert at end "NSIsLogLevelEnabled% (" & u% & ", " && logger$ & ") =" && ret% to
LB_RESULT
endFor
```

Renverra comme résultat :

```
NSIsLogLevelEnabled% (1, fr.natsystem.myAppli) = 0 ; (TRACE false%)
NSIsLogLevelEnabled% (2, fr.natsystem.myAppli) = 0 ; (DEBUG false%)
NSIsLogLevelEnabled% (3, fr.natsystem.myAppli) = 1 ; (INFO true%)
NSIsLogLevelEnabled% (4, fr.natsystem.myAppli) = 1 ; (WARNING true%)
NSIsLogLevelEnabled% (5, fr.natsystem.myAppli) = 1 ; (ERROR true%)
NSIsLogLevelEnabled% (6, fr.natsystem.myAppli) = 1 ; (FATAL true%)
```

Constantes_LL_* (Librairie nclLogger)

Ces constantes sont utilisées par la fonction nsIsLogLevelEnabled%

Il existe 6 niveau de log utilisables dans la librairie nclLOGGER

du moins important au plus critique : TRACE, DEBUG, INFO, WARN, ERROR, FATAL

le constantes associées sont :

LL_TRACE% (1) Log au niveau TRACE

LL_DEBUG% (2) Log au niveau DEBUG

LL_INFO% (3) Log au niveau INFO

LL_WARN% (4) Log au niveau WARNING

LL_ERROR% (5) Log au niveau ERROR

LL_FATAL% (6) Log au niveau FATAL

Note: en général, pour la mise au point des applications, les niveaux TRACE et DEBUG sont utilisés.

Voir aussi Fonction `nsIsLogLevelEnable%`,

Exemple nclLOGR (Librairie nclLOGR)

Voici un exemple de fichier NSLOGR.INI permettant d'effectuer une trace simple dans le fichier (TMP)\natlog\natsys.log

```
# nclLogr definition

# Trace Levels
# TRACE,DEBUG,INFO,WARN,ERROR,FATAL

#####
# Define file loggers
#####
# Applicative logger, using nclLogr APIs
log4j.logger.fr.natsystem.myAppli=INFO, myAppliLog

# Applicative logger, using NS_TRACE or THINGS_TRACE
log4j.logger.fr.natsystem.logger.trace=TRACE, myAppliLog

# internal Nat System Logger
log4j.logger.fr.natsystem.logger.internal=INFO, myAppliLog

#####
# Define file appender & layout
#####
log4j.appender.myAppliLog=org.apache.log4j.RollingFileAppender
log4j.appender.myAppliLog.FileName=${TMP}\natlog\natsys.log
log4j.appender.myAppliLog.layout=org.apache.log4j.PatternLayout
log4j.appender.myAppliLog.layout.ConversionPattern=%d{ISO8601} [%t] %p %c %m%n
```

Exemple d'utilisation

si dans le fichier NSLOGR.INI, on a la définition suivante,

le code NCL suivant

```
local logger$
logger$ = "fr.natsystem.myAppli"

nsLogTrace (logger$, "test Log au niveau Trace")
nsLogDebug (logger$, "test Log au niveau Debug a l'emplacement " & #location)
nsLogInfo (logger$, "test Log au niveau Info a l'emplacement " & #location)
NS_TRACE ("test NS_TRACE a l'emplacement " & #location)
```

Renverra comme résultat :

```
2023-06-13 15:36:27,632 [0x00002408] TRACE fr.natsystem.myAppli test Log au niveau Trace
2023-06-13 15:36:27,634 [0x00002408] DEBUG fr.natsystem.myAppli test Log au niveau Debug a
l'emplacement W.W_NSLOGGER.PB_LOG_DOC.EXECUTED(6)
2023-06-13 15:36:27,634 [0x00002408] INFO fr.natsystem.myAppli test Log au niveau Info a
l'emplacement W.W_NSLOGGER.PB_LOG_DOC.EXECUTED(7)
2023-06-13 15:36:27,634 [0x00002408] TRACE fr.natsystem.logger.trace test NS_TRACE a
l'emplacement W.W_NSLOGGER.PB_LOG_DOC.EXECUTED(8)
```

Actualisation quotidienne du nom du fichier log

en rajoutant les deux lignes suivantes dans le fichier NSLOGR.INI, le fichier de log est archivé quotidiennement avec la date à la suite du nom

```
log4j.appender.myAppliLog.rollingPolicy = org.apache.log4j.rolling.TimeBasedRollingPolicy
log4j.appender.myAppliLog.rollingPolicy.FileNamePattern =
${TMP}\natlog\natsys%d{yyyy_MM_dd}.log
```

Voir aussi : [Librairie NCLLOGR](#)

Compatibilité avec l'ancienne librairie NSLOGR.NCL

La librairie **NSLOGR.NCL** est remplacée par **nclLogr.NCL** qui met à disposition un jeu d'instruction plus simple et efficace.

Pour les utilisateurs de l'ancienne librairie NSLOGR.NCL, vous avez deux possibilités :

- Désactiver le nouveau mécanisme en positionnant initLogger à FALSE dans le fichier nslib.ini :

```
[Miscellaneous]
initLogger=False
```

dans le fichier nslib.ini : votre application log comme auparavant.

- Supprimer votre initialisation NSLOGR.NCL, et laisser le runtime prendre en charge l'initialisation du mécanisme de log (il faudra peut-être renommer le fichier ini du logger et le compléter).
Votre ancienne log vient s'ajouter au nouveau.

LIBRAIRIE NSCRYPT

Cette librairie permet l'utilisation de **fonction de hachage** ainsi qu'une **fonction de dérivation de clé (PBKDF2)**.

Référence de la librairie nsCrypt

Référence de la librairie nsCrypt

Voici la liste des fonctions de la librairie **nsCrypt** regroupées par catégories.

Constantes des fonctions de hachage et des erreurs

Constantes HASH *%

Liste des fonctions de la librairie NSCRYPT

Fonction de hash

Fonction de dérivation de clé

Fonction de récupération des erreurs lors du hash ou de la dérivation de clé

Fonction de lecture de message d'erreur

Exemple d'utilisation

Exemple nsCrypt

Fonction **NS_HASH_STR\$** (Librairie NsCrypt)

Hache du texte passé en paramètre selon la fonction de hachage choisie.

Syntaxe	NS_HASH\$ (<i>dynstr text\$, hash_type%</i>)			
Paramètres	text\$	dynstr		Text a hacher
	charset%	Int		Type d'algorithme de hachage (HASH_TYPE_*)
	hash_type%	Integer		Charset cible des caractères hachés (CHS_* ou -1)

Valeur retournée	dynstr	Renvoie la valeur (hexadécimal) hachée selon le type d'algorithme de hachage.
-------------------------	---------------	---

Remarque:

- Les erreurs de **NS_HASH_STR\$** sont récupérable par la fonction **NS_HASH_GET_ERROR%**.
- Le parameter *charset%* indique le charset (voir les constantes CHS_* dans nsmisc.ncl) cible vers lequel les caractères de la chaîne doivent être traduit (à partir du charset courant du projet). La valeur -1 indique de ne pas faire de traduction, sauf pour la cible Java où les chaînes sont stockées en Unicode et doivent être traduites en octets (byte[]) avant hachage, dans ce cas, -1 indique qu'il faut les traduire vers le charset courant du projet. L'utilisation de CHS_UTF8 peut être une option intéressante dans le cas où la valeur hachée doit pouvoir être comparée (ou stockée en base) avec ce qu'elle serait sur des systèmes utilisant l'UTF-8 par défaut (comme Linux).
- **En cas d'erreur** (NS_HASH_GET_ERROR% < 0) la fonction **NS_HASH\$** retourne une chaîne vide "
- **En cas de warning** (NS_HASH_GET_ERROR% > 0) la fonction **NS_HASH\$** retourne bien le hash du texte

Voici la liste des **types d'algorithme de hachage disponible** :

- **MD5**
- **SHA-256**
- **SHA-512**

Exemple

```
local DynStr entry$, DynStr result$, err%

entry$ = "Bonjour"

; remarque en cas de warning NS_HASH_STR$ retourne le résultat
; remarque en cas d'erreur NS_HASH_STR$ retourne une chaîne vide ''
result$ = NS_HASH_STR$(entry$, HASH_TYPE_SHA512%, -1)

err% = NS_HASH_GET_ERROR%
if err% <> 0
    message " Error n " & err%, NS_HASH_GET_ERROR$(err%)
endif
```

Renvoie:

```
'c447dff0d671f62ad580b255b64f7a8f6a30d1b828569cee08b7c861239f8d4856ef38a116671
8b045a9713876336c1f623619f6a78fc891d48d0b98c703def3'
```


Voir aussi : [Librairie nsCrypt](#) , [NS_HASH_BYTES\\$](#) , [NS_HASH_PKCS5_PBKDF2_HMAC\\$](#) , [NS_GET_HASH_ERROR\\$](#) , [NS_GET_HASH_ERROR_MESSAGE\\$](#) , [Constantes HASH](#) %

Fonction [NS_HASH_PKCS5_PBKDF2_HMAC\\$](#) (Librairie NsCrypt)

Renvoie une clé PBKDF2 dérivée du mot de passe fourni selon la fonction de hachage choisie ainsi que son salage.

Syntaxe	NS_HASH_PKCS5_PBKDF2_HMAC\$ \$(DynStr passwd\$, DynStr hexSalt\$, iterations%, hash_type%)			
Paramètres	passwd\$	dynstr	I	La chaîne à partir de laquelle il faut dériver la clé
	hexSalt\$	dynstr	I	Octets de « salage » (suite ininterrompue de paires de chiffres hexadécimaux).
	iterations%	Int	I	Le nombre d'itération de dérivation de clé recommandé est 1000 itérations (RFC-2898)
	hash_type%	Int	I	Type d'algorithme de hachage (HASH_TYPE_*)
Valeur retournée	dynstr	La valeur hachée (hexadécimal) de la clé PBKDF2 selon le type d'algorithme de hachage choisi.		

Remarque:

- Les erreurs de **NS_HASH_PKCS5_PBKDF2_HMAC \$** sont récupérable par la fonction **NS_HASH_GET_ERROR%**.
- En cas d'erreur (**NS_HASH_GET_ERROR% < 0**) la fonction **NS_HASH_PKCS5_PBKDF2_HMAC\$** retourne une chaîne vide "
- En cas de warning (**NS_HASH_GET_ERROR% > 0**) la fonction **NS_HASH_PKCS5_PBKDF2_HMAC\$** retourne bien la dérivation de clé du mot de passe

Voici la liste des **types d'algorithme de hachage disponible** :

- SHA-256**
- SHA-512**

Exemple

```
local DynStr password$, DynStr result$
local DynStr sel$
local nb_iterations%

password$ = 'Bonjour'

; 32 octets aléatoires généré par https://www.random.org/bytes/,
; espaces et sauts de lignes retirés.
```

```

sel$ = '2830c2a6d836f5a274694352fad5c094569868b0331c41fc5721189a1b3efcc1'
nb_iterations%=1600 ; recommandation de mini 1000 (rfc 2898)

; remarque en cas de warning NS_HASH_PKCS5_PBKDF2_HMAC$ retourne le resultat
; remarque en cas d'erreur NS_HASH_PKCS5_PBKDF2_HMAC$ retourne une chaîne vide ''
result$ = NS_HASH_PKCS5_PBKDF2_HMAC$(password$, sel$, nb_iterations%, HASH_TYPE_SHA512%)

err% = NS_HASH_GET_ERROR%
if err% <> 0
    message " Error n " & err%, NS_HASH_GET_ERROR$(err%)
endif

```

Renvoie (dans result\$) :

'3c15f1df766b25ffe93c2b7b7ece2c5395109a1ae6400dff67f243f4227804a79fcf59f41c493e9650d5b5bc0a752ebf9a778a3d63f90d0551229f0a4a15f0c6'

Voir aussi : Librairie nsCrypt, NS_HASH_STR\$, NS_HASH_BYTES\$, NS_GET_HASH_ERROR%, NS_GET_HASH_ERROR MESSAGE\$, Constantes HASH *%

Fonction NS_GET_HASH_ERROR% (Librairie NsCrypt)

Renvoie un code erreur en cas d'erreur sur la fonction de hachage (**NS_HASH\$**) ou celle de dérivation de clé (**NS_HASH_PKCS5_PBKDF2_HMAC\$**).

Syntaxe	NS_GET_HASH_ERROR% ()	
Valeur retournée	Int	Renvoie un code erreur, HASH_ERROR_*% ou HASH_WARN_*%.

1. La fonction **NS_GET_HASH_ERROR%** retourne donc :

- a)** 0 s'il n'y a eu aucune erreur,
- b)** un nombre positif pour toute erreur non fatale (l'instruction a été exécutée, mais émet un Warning),
- c)** un nombre négatif pour toute erreur fatale (l'instruction n'a pu être exécutée).

Exemple

```

local DynStr entry$, DynStr result$, err%

entry$ = "Bonjour"

; remarque en cas de warning NS_HASH_STR$ retourne le résultat
; remarque en cas d'erreur NS_HASH_STR$ retourne une chaîne vide ''
result$ = NS_HASH_STR$(entry$, HASH_TYPE_SHA512%, -1)

err% = NS_HASH_GET_ERROR%
if err% <> 0
    message " Error n " & err%, NS_HASH_GET_ERROR$(err%)
endif

```

Voir aussi : [Librairie nsCrypt](#) , [NS_HASH_STR\\$](#) , [NS_HASH_BYTES\\$](#) , [NS_HASH_PKCS5_PBKDF2_HMAC\\$](#) , [NS_GET_HASH_ERROR_MESSAGE\\$](#) , [Constantes HASH_*](#)

Fonction NS_GET_HASH_ERROR_MESSAGE\$ ([Librairie nsCrypt](#))

Retourne le texte du message d'erreur demandé.

Syntaxe	NS_GET_HASH_ERROR_MESSAGE\$ (<i>error%</i>)		
Paramètres	error%	Int	Code d'erreur
Valeur retournée	dynstr	Retourne le message d'erreur correspondant à error% . Voir NS_GET_HASH_ERROR% .	

La fonction **NS_GET_HASH_ERROR_MESSAGE\$** retourne donc le message d'erreur selon la valeur de **NS_GET_HASH_ERROR%**

Exemple

```
local DynStr entry$, DynStr result$, err%

entry$ = "Bonjour"

; remarque en cas de warning NS_HASH_STR$ retourne le résultat
; remarque en cas d'erreur NS_HASH_STR$ retourne une chaîne vide ''
result$ = NS_HASH_STR$(entry$, HASH_TYPE_SHA512%, -1)

err% = NS_HASH_GET_ERROR%
if err% <> 0
    message " Error n " & NS_HASH_GET_ERROR% , NS_HASH_GET_ERROR$(NS_HASH_GET_ERROR%)
endif
```

Voir aussi : [Librairie nsCrypt](#) , [NS_HASH_STR\\$](#) , [NS_HASH_BYTES\\$](#) , [NS_HASH_PKCS5_PBKDF2_HMAC\\$](#) , [NS_GET_HASH_ERROR\\$](#) , [Constantes HASH_*](#)

Constantes NSCRYPT ([Librairie nsCrypt](#))

Constantes des différentes **fonctions de hachage supportée** par la fonction **NS_HASH_STR\$**:

<u>Nom de l'algorithme</u>	<u>Famille</u>	<u>Constante associée</u>
MD5	Merkle–Damgård	HASH_TYPE_MD5%

SHA-256	SHA-2	HASH_TYPE_SHA256%
SHA-512	SHA-2	HASH_TYPE_SHA512%

Constantes des différentes **fonctions de hachage supportée** par la fonction **NS_HASH_PKCS5_PBKDF2_HMAC\$**:

Nom de l'algorithme	Famille	Constante associée
SHA-256	SHA-2	HASH_TYPE_SHA256%
SHA-512	SHA-2	HASH_TYPE_SHA512%

Constantes des différentes erreurs:

Nom de constante	Messages d'erreur	Cause	Valeur
	""	Pas d'erreur, l'exécution s'est bien déroulée.	0
HASH_ERROR_WRONG_ITERATIONS%	"Invalid parameter. Iterator must be greater than 0."	Erreur : le nombre d'itération doit être supérieur à 0.	-1
HASH_ERROR_WRONG_HASH_TYPE%	"Invalid parameter. Incorrect hash type"	Erreur : cet algorithme de hachage n'est pas valide.	-2
HASH_ERROR_WRONG_SALT%	"NS_HASH_PKCS5_PBKDF2_HMAC invalid parameter: salt\$ cannot be empty and must be unseparated pairs of hexadecimal digits"	Erreur : le sel ne peut pas être vide.	-3
HASH_ERROR_MD5_NOT_AVAILABLE%	NS_HASH_PKCS5_PBKDF2_HMAC invalid parameter: HASH_TYPE_MD5% is not supported for this function."	Erreur : Le type d'algorithme de hachage MD5 n'est pas supporté par la fonction NS_HASH_PKCS5_PBKDF2_HMAC .	-4

Constantes des différents warning:

Nom de constante	Messages d'erreur	Cause	Valeur
	""	Pas d'erreur, l'exécution s'est bien déroulée.	0
HASH_WARN_EMPTY_TEXT%	"Unexpected parameter value: text\$ empty."	Alerte: sur le fait que le texte a hacher est vide.	+1
HASH_WARN_EMPTY_PASSWORD%	"NS_HASH_PKCS5_PBKDF2_HMAC unexpected parameter value: password\$ empty."	Alerte: sur le fait que le mot de passe a hacher est vide.	+2

Voir aussi : [Librairie nsCrypt](#) , [NS_HASH_BYTES\\$](#) , [NS_HASH_STR\\$](#) , [NS_HASH_PKCS5_PBKDF2_HMAC\\$](#) , [NS_GET_HASH_ERROR%](#) , [NS_GET_HASH_ERROR_MESSAGE\\$](#)

Exemple nsCrypt ([Librairie nsCrypt](#))

Exemple d'utilisation du **NS_HASH\$**

```
local DynStr entry$, DynStr result$, err%
entry$ = 'Bonjour'
result$ = NS_HASH_STR$(entry$, HASH_TYPE_SHA512%, -1)
err% = NS_HASH_GET_ERROR%
if err% <> 0
    message " Error n " & NS_HASH_GET_ERROR% , NS_HASH_GET_ERROR$(NS_HASH_GET_ERROR%)
endif
```

Renvoie :

```
'c447dff0d671f62ad580b255b64f7a8f6a30d1b828569cee08b7c861239f8d4856ef38a1166718b045a9713876336c1f623619f6a78fc891d48d0b98c703def3'
```

Exemple d'utilisation de **NS_HASH_BYTES\$**

```
local bytes%(1)[8], DynStr result$, err%
bytes%[0] = $a1
bytes%[1] = $c9
bytes%[2] = $39
```

```

bytes%[3] = $1f
bytes%[4] = $b2
bytes%[5] = $cc
bytes%[6] = $19
bytes%[7] = $be

; remarque en cas de warning NS_HASH_BYTES$ retourne le résultat
; remarque en cas d'erreur NS_HASH_BYTES$ retourne une chaîne vide ''
result$ = NS_HASH_BYTES$(bytes%, sizeof bytes%, HASH_TYPE_SHA512%)

err% = NS_HASH_GET_ERROR%
if err% <> 0
    message " Error n " & err%, NS_HASH_GET_ERROR$(err%)
endif

```

Renvoie (dans result\$) :

'e14d510d87697c2d737af69f2f2dcc2c683468fd98645772487274447fb95bfa04f050c57f89770ded21189822159b9bf46c2d4c2c7515b460ce724a61108a1d'

Exemple d'utilisation du **NS_HASH_PKCS5_PBKDF2_HMAC\$**

```

local DynStr password$, DynStr result$
local DynStr sel$
local nb_iterations%

password$ = 'Bonjour'

; 32 octets aléatoires généré par https://www.random.org/bytes/,
; espaces et sauts de lignes retirés.
sel$ = '2830c2a6d836f5a274694352fad5c094569868b0331c41fc5721189a1b3efcc1'
nb_iterations%=1600 ; recommandation de mini 1000 (rfc 2898)

; remarque en cas de warning NS_HASH_PKCS5_PBKDF2_HMAC$ retourne le resultat
; remarque en cas d'erreur NS_HASH_PKCS5_PBKDF2_HMAC$ retourne une chaîne vide ''
result$ = NS_HASH_PKCS5_PBKDF2_HMAC$(password$, sel$, nb_iterations%, HASH_TYPE_SHA512%)

err% = NS_HASH_GET_ERROR%
if err% <> 0
    message " Error n " & err%, NS_HASH_GET_ERROR$(err%)
endif

```

Renvoie :

'7152d953007e05485201d26c89df801ae771d625e21df2dc4b524d8acf9404c216569806264f05e63cf3bdb6ab54d7ff4970b2b9a22ebf5122f51f796f752871'

Voir aussi : Librairie nsCrypt , NS_HASH_STR\$, NS_HASH_BYTES\$, NS_HASH_PKCS5_PBKDF2_HMAC\$, NS_GET_HASH_ERROR% , NS_GET_HASH_ERROR_MESSAGE\$, Constantes HASH %%

GLOSSAIRE

A

Attribut: Un attribut XML permet de spécifier les caractéristiques d'un élément. Par exemple, on peut définir les caractéristiques d'une voiture par sa marque et sa couleur sous la forme : `<Voiture Marque="Toyota" Couleur="Rouge"/>` Syntaxe : `<nom_élément nom_attribut1= "valeur de l'attribut1" nom_attribut2= "valeur de l'attribut2">`

C

CDATASection: Les sections CDATA sont utilisées pour protéger des caractères de balisage dans des portions de textes. Le seul délimiteur reconnu dans une section CDATA est la chaîne "]]>" qui ferme les sections CDATA. Les sections CDATA ne peuvent pas être imbriquées les unes dans les autres. L'objectif premier est d'inclure tel quel des fragments XML, sans avoir à protéger tous les délimiteurs propres au langage.

D

Document: Un nœud Document représente tout le document XML et constitue donc la racine de l'arbre.

DocumentFragment: Un nœud DocumentFragment représente un fragment du Document.

DocumentType: Chaque Document a un attribut doctype dont la valeur est soit null soit un nœud DocumentType. Le DocumentType permet de définir le type du Document.

DOM: DOM (Document Object Model) correspond à un langage standard d'interface (API) permettant à un programme de naviguer dans un arbre XML et d'en lire ou d'en modifier le contenu. DOM impose de construire un arbre en mémoire contenant l'intégralité des éléments du document en mémoire.

DTD: La DTD (Document type Definition) est un composant optionnel d'un document XML. Il permet de différencier un document bien formé d'un document valide. Un document bien formé respecte les règles syntaxiques du langage XML. Un document valide est conforme à la grammaire définie par la DTD. Ainsi, associer une DTD à un document XML permet de vérifier sa validité. La DTD spécifie : "Les balises contenues dans les documents, "Les balises pouvant contenir d'autres balises, "Le nombre et la séquence des balises, "Les attributs joints aux balises, "La valeur des attributs.

E

Element: Un nœud Element correspond à l'objet de base d'un document XML. Syntaxe : `<nom_élément nom_attribut1= "valeur de l'attribut1" nom_attribut2= "valeur de l'attribut2"> données ...</nom_élément>`

Entity: Un nœud Entity représente n'importe quelle entité d'un document XML qu'elle soit analysable ou non par le parseur. Notez que l'interface modélise l'entité elle-même et non sa déclaration. La modélisation des déclarations d'entités sera intégrée dans une prochaine spécification de DOM.

Entity reference: Les nœuds EntityReference correspondent à des abréviations pour des données répétitives. Elles sont référencées dans le document XML et délimitées par les caractères & et ; . Exemple de déclaration d'entité : `<!ENTITY xml " Extensible Markup Language ">` Exemple d'utilisation dans un document XML : `Le langage XML (&xml ;)` ... Résultat : `Le langage XML (Extensible Markup Language)` ...

Espace de noms: Un espace de noms est un mécanisme utilisé pour lever les éventuelles ambiguïtés des balises. Par exemple, la balise `<NOM>` peut désigner le nom d'une personne ou d'un produit. Les espaces de noms permettent de les différencier en utilisant un préfixe : `" perso:nom "` et `" produits:nom "`. Exemple de déclaration des espaces de noms : `<BIBLIO xmlns= " http://www.purchase.com " xmlns:perso= " http://www.noms.org ">` Les balises non préfixées se rapporteront à l'espace de noms par défaut `" http://www.purchase.com "` . Les balises préfixées par `" perso : "` se rapporteront à l'URL `" http://www.noms.org "`.

F

Feuille de style: Une feuille de style contient un ensemble de règles de présentation d'un document.

N

Namespaces: Un espace de noms est un mécanisme utilisé pour lever les éventuelles ambiguïtés des balises. Par exemple, la balise `<NOM>` peut désigner le nom d'une personne ou d'un produit. Les espaces de noms permettent de les différencier en utilisant un préfixe : `" perso:nom "` et `" produits:nom "` Exemple de déclaration des espaces de noms : `<BIBLIO xmlns= " http://www.purchase.com " xmlns:perso= " http://www.noms.org ">` Les balises non préfixées se rapporteront à l'espace de noms par défaut `" http://www.purchase.com "` . Les balises préfixées par `" perso : "` se rapporteront à l'URL `" http://www.noms.org "`.

Noeud: Un nœud correspond au point d'un arbre d'où partent une ou plusieurs branches.

Notation: Un nœud Notation sert, soit pour déclarer par un nom, le format d'une entité non contrôlée par le parseur, soit pour des déclarations formelles de cibles d'instructions de traitement. Un nœud Notation n'a aucun parent. Syntaxe pour définir le type de données non XML utilisé par une entité externe : `< !NOTATION NomNotation SYSTEM|PUBLIC " ValeurNotation ">` `< !ENTITY NomEntiteExterne SYSTEM " FichierNonXML " NDATA NomNotation>` Exemple : `< !NOTATION JPEG SYSTEM " PaintShopPro.exe ">` `< !ENTITY logo SYSTEM " Nat System.jpg " NDATA JPEG >`

P

Parseur: Un parseur XML analyse les flux XML et stocke les contenus en mémoire sous forme de DOM (Document Object Model) pour des traitements ultérieurs.

Processing Instruction: Un nœud ProcessingInstruction représente une instruction de traitement, utilisée en XML comme un moyen pour conserver des informations spécifiques à un processeur dans le texte même du document. Les instructions de traitement commençant par xml ont un usage réservé par le standard XML. Syntaxe : `< ? Nom_Instruction ContenuLibre ?>` Exemple de la déclaration XML : `< ?xml version=" 1.0 " encoding=" UTF-8 " standalone=" yes" ?>`

S

SAX: SAX (Simple API pour XML) est une API basée sur un modèle événementiel, cela signifie que SAX permet de déclencher des événements au cours de l'analyse du document XML. Une application basée sur SAX peut gérer uniquement les éléments dont elle a besoin sans avoir à construire en mémoire une structure contenant l'intégralité du document.

W

W3C: Organisme de proposition et de normalisation des protocoles et langages du Web. Site du W3C : <http://www.w3.org>

X

XML: XML est un langage de balisage extensible, c'est-à-dire que les développeurs peuvent créer leurs propres balises en fonction du type d'information à stocker et/ou à échanger. XML distingue bien les données et leur présentation. La présentation des données étant prise en charge par XSL.

XSL: XSL correspond au langage de feuilles de style XML. XSL définit différentes règles de présentation d'un document XML selon le support de lecture.

XSLT: Le processus XSLT permet de transformer un document XML en appliquant la feuille de style adéquate selon le type de terminal connecté.

INDEX

A

ACS_AUTOPLAY 248
 ACS_CENTER 248
 ACS_TIMER 248
 ACS_TRANSPARENT 248
 ACTIVATEEDITINPLACE 146
 ADJUST_MOVE% 137
 ADJUST_SIZE% 137
 ADJUSTLISTBOXCOLUMNS 144
 ARCTAN# 36
 ASCII2EBCDIC\$ 181

B

BEGINEDITINPLACE 148
 BMPF_BTNBORDERMASK% 193
 BMPF_DEFBTNBORDER% 193
 BMPF_HAVEBTNBORDER% 193
 BMPF_NOBTNBORDER% 193

C

CALL_PREVCALLBACK 209
 CHANGEWINDOWSTYLE 201
 CHDIR 80
 CHDISK 82
 CHK_ENDSESSION% 134
 CHK_LOSEFOCUS% 134
 CHK_PUSHBUTTON% 134
 CHS_ANSI% 197
 CHS_ASCII% 197
 CHS_ASCHIIMAC% 197
 CHS_EBCDIC% 197
 CHS_EBCDIC_037 197
 CHS_EBCDIC_273 197
 CHS_EBCDIC_280 197
 CHS_EBCDIC_284 197
 CHS_EBCDIC_285 197
 CHS_EBCDIC_297 197

CHS_EBCDIC_500 197
 CHS_HPROMAN8 197
 CHS_ISO8859_1 197
 CHS_ISO8859_15 197
 CHS_LATIN1 197
 CHS_PC437 197
 CHS_PC850 197
 CHS_ROMAN 197
 CHS_UTF8 197
 CKM_CB% 132
 CKM_CBE% 132
 CKM_CK% 132
 CKM_DEF% 132
 CKM_EF% 132
 CKM_FCBE% 132
 CKM_FEF% 132
 CKM_LB% 132
 CKM_MLE% 132
 CKM_PB% 132
 CKM_RB% 132
 CLEANTMPDIR 169
 CLOSECONSOLE% 44
 COMPILEREGREXPR% 190
 CON_NFY_% 44
 CONCATPATH1\$ 88
 CONCATPATH2\$ 88
 CONCATPATH3\$ 88
 CONCATPATH4\$ 88
 Constante DI_WINCMNCTRL% 246
 Constantes DCA_BMP_* 244
 Constantes DCA_CB_* 241
 Constantes DCA_CHK_* 242
 Constantes DCA_EF_* 237
 Constantes DCA_GB_* 245
 Constantes DCA_HS_* 246
 Constantes DCA_LB_* 239

Constantes DCA_MLE_* 240
 Constantes DCA_MNU_* 246
 Constantes DCA_RB_* 242
 Constantes DCA_ST_* 243
 Constantes DCA_VS_* 245
 Constantes GTE_*% 173
 Constantes NS_AS* 173
 Constantes SBVM_*% 124
 Constantes SMIP_*% 166
 CONTROLNAME\$ 121
 COS# 36
 CURRENCY\$ 183
 CURRENTDATE% 7
 CURRENTDAY% 7
 CURRENTTIME% 7
 D
 DA_APPEND% 289
 DA_CAPACITY% 289
 DA_CLEAR% 290
 DA_CONTAINS% 290
 DA_CREATE 291
 DA_CREATEWITHCAPACITY 292
 DA_DELETE% 292
 DA_GETAT 293
 DA_GETFIRST 293
 DA_GETLAST 293
 DA_INDEXOF% 294
 DA_INSERTARRAYAT% 294
 DA_INSERTAT% 295
 DA_INSERTDYNARRAYAT% 296
 DA_ISEMPY% 296
 DA_LASTINDEXOF% 297
 DA_REMOVE% 297
 DA_REMOVEAT% 298
 DA_RESERVE% 298
 DA_SETAT 299
 DA_SIZE% 299
 DA_SORT 300
 DATE\$ 25
 DATE% 26
 DATE_ERROR% 29
 Dates 5
 CURRENTDATE% 5
 DATE\$ 5
 DATE% 5
 DATE_ERROR% 5
 DD_MM_YY*% 5
 DD_MM_YYYY% 5
 GETDATEFORMAT% 5
 GETDATESEPARATOR\$ 5
 ISDATE% 5
 MM_DD_YY% 5
 SETDATEFORMAT 5
 SETDATESEPARATOR 5
 YY_MM_DD*% 5
 DC_ControlProperties 256
 DC_CreateControl% 261
 DC_DestroyControl% 261
 DC_GetControlProperties 263
 DC_GetControlPropertiesFromID 262
 DCA_BMP_ADJUSTSIZE 244
 DCA_BMP_CHECKBOX 244
 DCA_BMP_DISABLE 244
 DCA_BMP_HIDDEN 244
 DCA_BMP_ICON 244
 DCA_BMP_NOCHECKING 244
 DCA_BMP_PUSHBUTTON 244
 DCA_CB_BELOW 241
 DCA_CB_DISABLE 241
 DCA_CB_HIDDEN 241
 DCA_CB_OVER 241
 DCA_CHK_3STATES 242
 DCA_CHK_AUTOSIZE 242
 DCA_CHK_BELOW 242
 DCA_CHK_DISABLE 242
 DCA_CHK_HIDDEN 242
 DCA_EF_AUTOSCROLL 237
 DCA_EF_AUTOTAB 237
 DCA_EF_BELOW 237
 DCA_EF_CENTER 237

DCA_EF_DATE 237
DCA_EF_DISABLE 237
DCA_EF_EXPLODING 237
DCA_EF_FULLTEXT 237
DCA_EF_HIDDEN 237
DCA_EF_HIDETEXT 237
DCA_EF_HORZSCROLL 237
DCA_EF_INTEGER 237
DCA_EF_LEFT 237
DCA_EF_LOCKTEXT 237
DCA_EF_MARGIN 237
DCA_EF_NOBLANKS 237
DCA_EF_NUMBER 237
DCA_EF_OVER 237
DCA_EF_REQUIRED 237
DCA_EF_RIGHT 237
DCA_EF_SKIPBLANKS 237
DCA_EF_TIME 237
DCA_EF_UPCASE 237
DCA_EF_VERTSCROLL 237
DCA_GB_BELOW 245
DCA_GB_BELOW1 245
DCA_GB_CENTER 245
DCA_GB_ERASERECT 245
DCA_GB_HALFTONE 245
DCA_GB_HIDDEN 245
DCA_GB_LEFT 245
DCA_GB_MNEMONIC 245
DCA_GB_OVER 245
DCA_GB_OVER1 245
DCA_GB_RIGHT 245
DCA_HS_DISABLE 246
DCA_HS_HIDDEN 246
DCA_LB_BELOW 239
DCA_LB_DISABLE 239
DCA_LB_HIDDEN 239
DCA_LB_HORZSCROLLBAR 239
DCA_LB_MULTIPLESEL 239
DCA_LB_NOADJUSTPOS 239
DCA_LB_OVER 239
DCA_LB_REALTIME 239
DCA_LB_USERDRAW 239
DCA_MLE_ACTIVETAB 240
DCA_MLE_BELOW 240
DCA_MLE_DISABLE 240
DCA_MLE_HIDDEN 240
DCA_MLE_LOCKTEXT 240
DCA_MLE_NOADJUSTPOS 240
DCA_MLE_OVER 240
DCA_MLE_UPCASE 240
DCA_MLE_WORDWRAP 240
DCA_MNU_CHECKED 246
DCA_MNU_DISABLED 246
DCA_MNU_HIDDEN 246
DCA_MNU_RIGHTSIDE 246
DCA_MNU_SEPARATOR 246
DCA_MNU_SUBMENU 246
DCA_RB_AUTOSIZE 242
DCA_RB_BELOW 242
DCA_RB_DISABLE 242
DCA_RB_HIDDEN 242
DCA_RB_OVER 242
DCA_ST_AUTOSIZE 243
DCA_ST_BELOW 243
DCA_ST_CENTER 243
DCA_ST_ERASERECT 243
DCA_ST_HALFTONE 243
DCA_ST_HIDDEN 243
DCA_ST_LEFT 243
DCA_ST_MARGIN 243
DCA_ST_MNEMONIC 243
DCA_ST_OVER 243
DCA_ST_RIGHT 243
DCA_ST_WORDWRAP 243
DCA_VS_DISABLE 245
DCA_VS_HIDDEN 245
DD_MM_YY*% 8
DD_MM_YYYY% 8
DEFBMPBUTTONIZED% 194
DEFBMPTRANSPARENT% 194

DEV_ADJUSTX 205
 DEV_ADJUSTY 205
 DI_BITMAP% 117
 DI_CHECKBOX% 117
 DI_CHECKBOX3% 117
 DI_COMBBOX% 117
 DI_COMBBOXE% 117
 DI_CONTROL% 117
 DI_CUSTOM% 117
 DI_ENTRYFIELD% 117
 DI_GROUPBOX% 117
 DI_HSCROLL% 117
 DI_ICON% 117
 DI_LISTBOX% 117
 DI_MENUITEM% 117
 DI_MLE% 117
 DI_PUSHBUTTON% 117
 DI_RADIOBUTTON% 117
 DI_STATICTEXT% 117
 DI_TREEBOX% 117
 DI_VSCROLL% 117
 DI_WCC_Animation% 117, 248
 DI_WCC_DateTimePicker% 117, 248
 DI_WCC_HotKey% 117, 250
 DI_WCC_ListView% 117, 250
 DI_WCC_MonthCalendar% 117, 252
 DI_WCC_ProgressBar% 117, 253
 DI_WCC_TrackBar% 117, 253
 DI_WCC_TreeView% 117, 254
 DI_WINCMNCTRL% 117, 246
 DISKFREE# 83
 DISKFREE% 82
 DISKSIZE# 84
 DISKSIZE% 84
 DLG_CHECKCONTROLFOCUS% 203
 DLG_CHECKCONTROLS 203
 DT_CDROM% 85
 DT_ERASERECT% 85
 DT_FIXED% 85
 DT_HALFTONE% 85
 DT_LEFT% 85
 DT_MNEMONIC% 85
 DT_RAMDISK% 85
 DT_REMOTE% 85
 DT_REMOVABLE% 85
 DT_RIGHT% 85
 DT_TOP% 85
 DT_VCENTER% 85
 DT_WORDWRAP% 85
 DTS_APPCANPARSE 248
 DTS_LONGDATEFORMAT 248
 DTS_RIGHTALIGN 248
 DTS_SHORTDATECENTURYFORMAT
 248
 DTS_SHORTDATEFORMAT 248
 DTS_SHOWNONE 248
 DTS_TIMEFORMAT 248
 DTS_UPDOWN 248
 E
 EARCTAN# 36
 EBCDIC2ASCII\$ 181
 ECOS# 37
 ECURRENCY\$ 183
 EEXP# 38
 EFRAC# 32
 EINT# 33
 ELN# 39
 ENVCOUNT% 64
 ENVSTR\$ 65
 EPI# 35
 EPOWER# 33
 ESIN# 37
 ESQR# 34
 ESQRT# 35
 ESTRING\$ 182
 EXEC_NFY_TERMINATE% 55
 EXECPROG 57
 EXECUTEREGREXPR% 192
 Exemple NSMAPS 278
 Exemples 220

EXP# 38

Extensions du langage NCL pour les
calculs mathématiques 31

ARCTAN# 31

COS# 31

EFAC# 31

ELN# 31

ESIN# 31

EXP# 31

FRAC# 31

INT# 31

LN# 31

PI# 31

POWER# 31

SIN# 31

SQR# 31

SQRT# 31

Extensions du langage NCL pour
l'exécution de programmes 41

CLOSECONSOLE% 41

CON_NFY_% 41

EXEC_NFY_TERMINATE% 41

FEXE_% 41

KILLEXECUTE 41

OPENCONSOLE% 41

STARTCONSOLEEXECEX% 41

STARTEXCUTE2% 41

STARTEXCUTEEX% 41

STOPCONSOLEEXEC% 41

STOPEXCUTE2 41

STOPEXCUTEEX 41

F

F_BLOCKREAD 106

F_BLOCKWRITE 107

F_BYTEREAD 110

F_BYTESEEK 113

F_BYTEWRITE 111

F_CLOSE 102

F_CREATE% 102

F_EOF% 114

F_ERROR% 115

F_LOADFILE\$ 116

F_LOCK 113

F_OPEN% 104

F_POS% 115

F_READ 108

F_ROOPEN% 105

F_SaveFile 117

F_SEEK 112

F_SIZE% 116

F_TRUNCATE 112

F_UNLOCK 114

F_WRITE 109

FCOPYFILE 68

FERASE 67

FEXE_% 45

FEXE_HIDDEN% 45

FEXE_MAXIMIZED% 45

FEXE_MINIMIZED% 45

FEXE_NORMAL% 45

FEXE_USEPOSITIONSIZE% 45

FEXPAND\$ 71

FFINDCLOSEEX 78

FFINDFIRST\$ 73

FFINDFIRSTEX\$ 76

FFINDNEXT\$ 74

FFINDNEXTEX\$ 77

FGETATTR% 69

FGETDIR\$ 72

FGETTEXT\$ 72

FGETNAME\$ 73

FGETSIZE% 70

FGETSIZEEX% 64, 70

FGETTIME 78

Fichier NSFEXE.NCL 41

Fichier NSMATH.NCL 31

Fichier NSMISC.NCL 57

FIND_% 75

FIND_ANYFILE% 75

FIND_ARCHIVE% 75

FIND_DIRECTORY% 75
 FIND_HIDDEN% 75
 FIND_ORED_ONLY% 75
 FIND_READONLY% 75
 FIND_RETURN_ATTR% 75
 FIND_RETURN_DATE% 75
 FIND_RETURN_SIZE% 75
 FIND_RETURN_TIME% 75
 FIND_SYSFILE% 75
 FIRSTCONTROL 119
 FOCUSCONTROL 129
 FOCUSCTRLATTRS 131
 Fonction NSMAPS_MAP_NEW 272
 Fonction NSMAPS_MAP_NEW2 274
 Fonction
 NSMAPS_MAP_STRING_NEW 276
 ForceDSLengh 224
 Formats de dates 215
 Formats d'heures 216
 Formats et spécifications 216
 Formats pour STRING\$ et ESTRING\$
 219
 FRAC# 32
 FRENAME 68
 FSETATTR 69
 FSETTIME 78
 G
 Gestion de repertoires temporaires
 167
 Gestion des List Box 143
 ACTIVATEEDITINPLACE 143
 ADJUSTLISTBOXCOLUMNS 143
 BEGINEDITINPLACE 143
 GETLISTBOXCELL 143
 QUERYEDITINPLACE 143
 TERMINATEEDITINPLACE 143
 GETBITMAPCONTROLSDEFAULTS%
 192
 GETCHECKMODE% 132
 GETCONTROLID% 127
 GETCONTROLKIND 204
 GETCONTROLSUBKIND 204
 GETCONTROLTEMPLATE 213
 GETCURRENTCHARSET% 197
 GETDATEFORMAT% 12
 GETDATEFORMAT2% 13
 GETDATESEPARATOR\$ 13
 GETDECIMALSEPARATOR\$ 188
 GETDESKTOPHEIGHT% 207
 GETDESKTOPWIDTH% 207
 GETDESKTOPX% 208
 GETDESKTOPY% 208
 GETDIR\$ 80
 GETDISK% 85
 GETDISKS% 86
 GETDISKTYPE% 86
 GETENV\$ 65
 GETESCAPECONTROL 134
 GetHeapArrayCount% 211
 GETHOSTCHARSET% 198
 GETLISTBOXCELL 145
 GETLONGDAY\$ 19
 GETLONGENV\$ 66
 GETLONGMONTH\$ 20
 GETMDISTR\$ 138
 GETMENUITEMPICTURE% 154
 GETMILESTONEYEAR% 29
 GETRETURNCONTROL 135
 GETSHORTDAY\$ 22
 GETSHORTMONTH\$ 24
 GETTABCONTROL 130
 GETTHOUSANDSSEPARATOR\$ 188
 GETTIMEAM\$ 16
 GETTIMEFORMAT% 15
 GETTIMEPM\$ 17
 GETTIMESEPARATOR\$ 18
 GETTMPDIR\$ 168
 GetTotalDynamicControls% 260
 GETVIRTUALKEYSTATE% 201
 GETVOLUMENAME\$ 87

GETWINDCAPTURE 136

Graphical Builder application
management 213, 215

H

HASH_ERROR_EMPTY_SALT% 317

HASH_ERROR_MD5_NOT_AVAILABLE
% 317

HASH_ERROR_WRONG_HASH_TYPE%
317

HASH_ERROR_WRONG_ITERATIONS%
317

HASH_TYPE_MD5% 317

HASH_TYPE_SHA256% 317

HASH_TYPE_SHA512% 317

HASH_WARN_EMPTY_PASSWORD%
317

HASH_WARN_EMPTY_TEXT% 317

Heures 6

CURRENTTIME% 6

GETTIMEAM\$ 6

GETTIMEFORMAT% 6

GETTIMEPM\$ 6

GETTIMESEPARATOR\$ 6

HH_MM% 6

HH_MM*% 6

HH_MM_AMPM% 6

HH_MM_SS% 6

HH_MM_SS_AMPM% 6

ISTIME% 6

SETTIMEAMPM. 6

SETTIMESEPARATOR 6

TIME\$ 6

TIME% 6

TIME_ERROR% 6

HH_MM% 9

HH_MM_AMPM% 9

HH_MM_SS% 9

HH_MM_SS_AMPM% 9

HideRemovedCmdLineParams% 62

I

ILD_MASK% 158

ILD_TRANSPARENT% 158

INSTALL_CALLBACK 208

INT# 32

ISDATE% 25

IsDSNull% 223

ISESCAPECONTROL% 138

ISRETORESCCONTROL% 138

ISRETURNCONTROL% 138

ISTIME% 25

ISVALIDSELFHANDLE% 129

ItExists% 260

J

Jours 5

CURRENTDAY% 5

GETLONGDAY\$ 5

GETSHORTDAY\$ 5

SETLONGDAY 5

SETSHORTDAY 5

K

KILLEXECUTE 54

L

LBCC_BELOWLASTLINE% 152

LBCC_CLIPALL% 152

LBCC_CLIPBOTTOM% 152

LBCC_CLIPLEFT% 152

LBCC_CLIPRIGHT% 152

LBCC_CLIPTOP% 152

LBCC_RIGHTOFLASTCOL% 152

LBCC_TOPLOCKED% 152

LBCF_CLIPCOORDS% 153

LBCF_ENLARGERIGHTCOL% 153

LBCF_FROMXY% 153

LBCF_WHOLELINE% 153

LGETENV 66

Librairie de dates et heures NSDate 5

Catégories fonctionnelles de la
librairie NSDate 5

- Extensions du langage NCL pour les dates et heures 5
- Fichier NSDATE.NCL 5
- Installation 5
- Référence de la librairie NSDate 5
- Librairie de fonctions diverses NSMisc 57
 - Catégories fonctionnelles de la librairie NSMisc 57
 - Fichier NSMISC.NCL 57
 - Installation 57
 - Référence de la librairie NSMisc 57
- Librairie de gestion de collection NSDYNCOLLEC 285
- Librairie de gestion de Map NSMAPS 265
 - Exemple NSMAPS 278
 - Référence de la librairie NSMAPS 265
- Librairie mathématique NSMath 31
 - Catégories fonctionnelles de la librairie NSMath 31
 - Extensions du langage NCL pour les calculs mathématiques 31
 - Fichier NSMATH.NCL 31
 - Installation 31
 - Référence de la librairie NSMath 31
- Librairie nsCrypt 313
- Librairie NSDYNCTR 237
- Librairie NSDynstr 223
 - Fichier NSDynstr.NCL 223
 - Installation 223
- Librairie NSFexe 41
 - Catégories fonctionnelles de la librairie NSFexe 41
 - Extensions du langage NCL pour exécution de programmes 41
 - Fichier NSFEXE.NCL 41
 - Référence de la librairie NSFexe 41
- librairie NSLOGR 303
- Librairie NSMLE 229
- LL_DEBUG% 309
- LL_ERROR% 309
- LL_FATAL% 309
- LL_INFO% 309
- LL_TRACE% 309
- LL_WARN% 309
- LN# 38
- LVS_ALIGNLEFT 250
- LVS_ALIGNMASK 250
- LVS_ALIGNTOP 250
- LVS_AUTOARRANGE 250
- LVS_EDITLABELS 250
- LVS_ICON 250
- LVS_LIST 250
- LVS_NOCOLUMNHEADER 250
- LVS_NOLABELWRAP 250
- LVS_NOSCROLL 250
- LVS_NOSORTHEADER 250
- LVS_OWNERDATA 250
- LVS_OWNERDRAWFIXED 250
- LVS_REPORT 250
- LVS_SHAREIMAGELISTS 250
- LVS_SHOWSELALWAYS 250
- LVS_SINGLESEL 250
- LVS_SMALLICON 250
- LVS_SORTASCENDING 250
- LVS_SORTDESCENDING 250
- LVS_TYPEMASK 250
- LVS_TYPESTYLEMASK 250
- M
- MakeCmdLineParam\$ 63
- MAKECONTROL 119
- Manipulation des listes 154
- Manipulation des listes d'images et affectation d'images aux items de menu 154
 - GETMENUITEMPICTURE% 154
 - SETMENUITEMPICTURE 154
 - WIL_APPENDFROMBMP% 154

WIL_DELETEPICTURES 154
 WIL_DRAWPICTURE 154
 WIL_FINDORADDONEPICTURE% 154
 WIL_FREEHASH 154
 WIL_GETCOUNT% 154
 WIL_GETSIZE 154
 WIL_SETCOUNT 154
 MB_*% 184
 MB_ABORT% 184
 MB_APPLICATION% 184
 MB_CANCEL% 184
 MB_DEFBUTTON1% 184
 MB_DEFBUTTON2% 184
 MB_DEFBUTTON3% 184
 MB_ENTER% 184
 MB_ERROR% 184
 MB_HELP% 184
 MB_ICONASTERISK% 184
 MB_ICONEXCLAMATION% 184
 MB_ICONHAND% 184
 MB_ICONQUESTION% 184
 MB_IGNORE% 184
 MB_MOVEABLE% 184
 MB_NO% 184
 MB_OK% 184
 MB_RETRY% 184
 MB_SYSTEM% 184
 MB_YES% 184
 MCS_DAYSTATE 252
 MCS_MULTISELECT 252
 MCS_NOTODAY 252
 MCS_NOTODAYCIRCLE 252
 MCS_WEEKNUMBERS 252
 MDI_ARRANGE_CASCADE% 141
 MDI_ARRANGE_HORZTILE% 141
 MDI_ARRANGE_VERTTILE% 141
 MDI_CANCEL% 139
 MDI_CASCADE% 139
 MDI_CLOSE% 139
 MDI_MAXIMIZE% 139
 MDI_MINIMIZE% 139
 MDI_MORE% 139
 MDI_MOVE% 139
 MDI_NEXT% 139
 MDI_RESTORE% 139
 MDI_SELECT% 139
 MDI_SIZE% 139
 MDI_TILE% 139
 MDIARRANGEWINDOWS 140
 MDISHOWHIDDENWINDOWS 141
 MESSAGE% 187
 MISCERROR% 90
 MKDIR 81
 MM_DD_YY% 8
 MM_DD_YYYY% 8
 MODE_FULL_SCREEN% 59
 MODE_INVISIBLE% 59
 MODE_MAXIMIZE% 59
 MODE_MINIMIZE% 59
 MODE_NOAUTOCLOSE% 59
 MODE_NORMAL% 59
 MODE_PM_SCREEN% 59
 MODE_USE_POSITION% 59
 MODE_VIO_SCREEN% 59
 Mois 6
 GETLONGMONTH\$ 6
 GETSHORTMONTH\$ 6
 SETLONGMONTH 6
 SETSHORTMONTH 6
 MyTrace 169
 N
 NEXTCONTROL 120
 NS_AS_BOTTOM 173
 NS_AS_BOTTOM_RIGHT 173
 NS_AS_HEIGHT_LEFT 173
 NS_AS_HEIGHT_RIGHT 173
 NS_AS_LEFT 173
 NS_AS_NONE 173
 NS_AS_NSDK_DEFAULT 173

NS_AS_PBOTTOM 173
 NS_AS_PLEFT 173
 NS_AS_PRIGHT 173
 NS_AS_PTOP 173
 NS_AS_RIGHT 173
 NS_AS_TOP 173
 NS_AS_TOP_LEFT 173
 NS_AS_TOP_RIGHT 173
 NS_AS_WIDTH 173
 NS_AS_WIDTH_BOTTOM 173
 NS_AS_WIDTH_HEIGHT 173
 NS_AS_WIDTH_TOP 173
 NS_GET_HASH_ERROR% 316
 NS_GET_HASH_ERROR_MESSAGE\$ 317
 NS_HASH\$ 313
 NS_HASH_PKCS5_PBKDF2_HMAC\$ 315
 NS_TRACE 170
 NSDate 5
 Librairie 5
 NSDYNCTR 237
 NSEXISTDIR% 79
 NSEXISTFILE% 79
 NSFexe 41
 Installation 41
 Librairie 41
 NSGETMENUWINDOW 128
 NSISLOGLEVELENABLE% 308
 NSLOGDEBUG 305
 NSLOGERROR 307
 NSLOGFATAL 308
 NSLOGINFO 305, 306
 NSLOGTRACE 304
 NSLOGWARN 307
 NSMAPS_MAP_CONTAINS% 277
 NSMAPS_MAP_DISPOSE 271
 NSMAPS_MAP_GET 269
 NSMAPS_MAP_GET_ITERATOR 268
 NSMAPS_MAP_GET_STRING 270
 NSMAPS_MAP_ITERATOR_DISPOSE 273
 NSMAPS_MAP_ITERATOR_GETKEY 273
 NSMAPS_MAP_ITERATOR_GETSTRING 268
 NSMAPS_MAP_ITERATOR_GETVALUE 267
 NSMAPS_MAP_ITERATOR_NEXT 272
 NSMAPS_MAP_PUT 276
 NSMAPS_MAP_REMOVE 266
 NSMAPS_MAP_SIZE 274
 NSMAPS_PUT_STRING 275
 NSMath 31
 NSMEMCOMP% 196
 NSMEMPOS% 125
 NSMisc 57
 Librairie 57
 NSMLE_CHANGE% 234
 NSMLE_CLEAR 230
 NSMLE_COPY 230
 NSMLE_CUT 229
 NSMLE_GETTEXT 231
 NSMLE_GETTEXTSIZE% 232
 NSMLE_ISCLIPBOARD% 233
 NSMLE_ISSELECTED% 232
 NSMLE_ISUNDO% 232
 NSMLE_LOAD% 230
 NSMLE_PASTE 230
 NSMLE_SEARCH% 234
 NSMLE_SELECTION\$ 233
 NSMLE_SETTEXT% 231
 NSMLE_UNDO 233
 NSSEARCHPATHS\$ 142
 NSSORT 175
 nsStrCmp% 210
 O
 OPENCONSOLE% 42
 P
 PARAMCOUNT% 61

PARAMSTR\$ 61
PBS_SMOOTH 253
PBS_VERTICAL 253
PI# 35
POWER# 33
PUTENV 67
Q
QUERYEDITINPLACE 150
R
RANDOM# 180
RANDOM% 180
RANDOMIZE 181
Référence de la librairie NSMAPS 265
REMOVE_CALLBACK 209
RemoveCmdLineParam 62
RMDIR 81
S
SBVM_ALL% 122
SBVM_BACK% 122
SBVM_CHANGE% 122
SBVM_EXACT% 122
SBVM_EXPTP% 122
SBVM_FIND% 122
SBVM_FWD% 122
SBVM_M_ACT% 122
SBVM_M_CMP% 122
SBVM_M_DIR% 122
SBVM_M_LBO% 122
SBVM_M_RNG% 122
SBVM_NOCASE% 122
SBVM_ONE% 122
SBVM_PART% 122
SBVM_REGEX% 122
SBVM_SEL% 122
SBVM_STRPOW% 122
SBVM_STRPBM% 122
SBVM_STRPPA% 122
SBVM_STRPTP% 122
SBVM_UNSEL% 122
SEG_EIPINFO 150
SEG_LBCELL 145
SEG_NSGENTRACE 172
SEG_SIZEORPOS 137
SELECTBYVALUE% 122
SELFFROMCONTROL% 128
SETBITMAPCONTROLSDEFAULTS 193
SETCHECKMODE 133
SETCURRENTCHARSET 198
SETDATEFORMAT 9
SETDATEFORMAT2 10
SETDATESEPARATOR 11
SETDECIMALSEPARATOR 188
SetDSNull 224
SETESCAPECONTROL 135
SETFOCUSCONTROLATTRIBUTES%
131
SETLONGDAY 18
SETLONGMONTH 20
SETMDISTR 140
SETMENUITEMPICTURE 155
SETMILESTONEYEAR 28
SetNSGenTrace 169
SETRETURNCONTROL 136
SETSHORTDAY 21
SETSHORTMONTH 23
SETTABCONTROL 130
SETTHOUSANDSSEPARATOR 189
SETTIMEAMPM 15
SETTIMEFORMAT 14
SETTIMESEPARATOR 17
SETZORDER 206
SHAREDMEMALLOCERROR% 194
SHORTENPATHNAME 126
SIN# 37
SMIP_EditCopy% 166
SMIP_EditCut% 166
SMIP_EditDelete% 166
SMIP_EditPaste% 166
SMIP_EditRedo% 166
SMIP_EditUndo% 166

SMIP_ExitApp% 166
 SMIP_FileClose1% 166
 SMIP_FileClose2% 166
 SMIP_FileDelete% 166
 SMIP_FileNew% 166
 SMIP_FileNew2% 166
 SMIP_FileOpen% 166
 SMIP_FilePrint% 166
 SMIP_FileSave% 166
 SMIP_FileSaveAll% 166
 SMIP_FindNext% 166
 SMIP_FindPrev% 166
 SMIP_FirstStdPict% 166
 SMIP_Help% 166
 SMIP_HelpContent% 166
 SMIP_HelpIndex% 166
 SMIP_HelpSearch% 166
 SMIP_Search% 166
 SMIP_Tools% 166
 SMIP_WinCascade% 166
 SMIP_WinChoose% 166
 SMIP_WinSplitHorz% 166
 SMIP_WinSplitVert% 166
 SMIP_Zoom% 166
 SQR# 34
 SQRT# 34
 STARTCONSOLEEXEC% 46
 STARTEXCUTE2% 51
 STARTEXCUTEEX% 49
 STOPCONSOLEEXEC% 48
 STOPEXCUTE2 54
 STOPEXCUTEEX 53
 STRING\$ 181
 T
 T_APPEND% 90
 T_CLOSE 91
 T_CREATE% 91
 T_EOF% 101
 T_EOL% 101
 T_ERROR% 102
 T_OPEN% 92
 T_READ# 96
 T_READ\$ 97
 T_READ% 96
 T_READLN# 96, 99
 T_READLN\$ 99
 T_READLN% 98
 T_READLNEX\$ 100
 T_WRITE 93
 T_WRITEEX 94
 T_WritelN 94
 T_WritelNEX 95
 TBS_AUTOTICKS 253
 TBS_BOTH 253
 TBS_BOTTOM 253
 TBS_DOWNSIZELEFT 253
 TBS_ENABLESELRANGE 253
 TBS_FIXEDLENGTH 253
 TBS_HORZ 253
 TBS_LEFT 253
 TBS_NOTHUMB 253
 TBS_NOTICKS 253
 TBS_REVERSED 253
 TBS_RIGHT 253
 TBS_TOOLTIPS 253
 TBS_TOP 253
 TBS_VERT 253
 TERMINATEEDITINPLACE 151
 TEST_MEMORY_ACCESS% 195
 TEST_READ_ACCESS 196
 TEST_READWRITE_ACCESS 195
 TIME\$ 27
 TIME% 28
 TIME_ERROR% 30
 TRANSLATECHARS 198
 TRANSLATECHARS2% 199
 TRANSLATECHS\$ 201
 TRANSP_BOTLEFT% 194
 TRANSP_BOTRIGHT% 194
 TRANSP_DEFAULT% 194

TRANSP_NEVER% 194
 TRANSP_TOLEFT% 194
 TRANSP_TOPRIGHT% 194
 TVS_CHECKBOXES 254
 TVS_DISABLEDRAHDROP 254
 TVS_EDITLABELS 254
 TVS_FULLROWSELECT 254
 TVS_HASBUTTONS 254
 TVS_HASLINES 254
 TVS_INFOTIP 254
 TVS_LINESATROOT 254
 TVS_NOHSCROLL 254
 TVS_NONEVENHEIGHT 254
 TVS_NOSCROLL 254
 TVS_NOTOOLTIPS 254
 TVS_RTLREADING 254
 TVS_SHOWSELALWAYS 254
 TVS_SINGLEEXPAND 254
 TVS_TRACKSELECT 254
 W
 WCCSEND% 177
 WCCSEND0% 177
 WCCSEND1% 178
 WCCSENDPTR% 179
 WCCSENDSTR% 179
 WIL_APPENDFROMBMP% 155
 WIL_APPENDFROMMEM% 165
 WIL_DELETEPICTURES 157
 WIL_DESTROY 157
 WIL_DRAWPICTURE 158
 WIL_FINDORADDONEPICTURE% 159
 WIL_FINDPICTURE% 161
 WIL_FREEHASH 162
 WIL_GETCOUNT% 163
 WIL_GETPICTURESCOUNT% 163
 WIL_GETSIZE 163
 WIL_SETCOUNT 164
 WINDOWFROMCONTROL% 127
 WINDOWNAME\$ 122
 WS_BORDER% 202
 WS_DLGBORDER% 202
 WS_HORZSCROLLBAR% 202
 WS_MENUBAR% 202
 WS_MINIMIZE% 202
 WS_MINMAX% 202
 WS_NOBYTEALIGN% 202
 WS_QUIT% 202
 WS_SIZEBORDER% 202
 WS_SYSMENU% 202
 WS_TITLE% 202
 WS_VERTSCROLLBAR% 202
 Y
 YY_MM_DD% 8
 YYYY_MM_DD% 8
 Z
 ZOK_BEHIND% 206
 ZOK_BOTTOM% 206
 ZOK_NOTOPMOST% 206
 ZOK_TOP% 206
 ZOK_TOPMOST% 206