
Comment utiliser cet Xtra

Pour pouvoir utiliser AbsFile, copier le fichier AbsFile.x32 dans le dossier "Xtras" du dossier d'installation de Director.

Pour obtenir un résumé des fonctionnalités de l'Xtra, tapez dans la fenêtre message de Director :
Put Xtra(« AbsFile »).interface().

Pour créer une instance de AbsFile, utilisez le code suivant :
Notation pointée (Director 7 et plus)
AbsObj=Xtra(« AbsFile »).new()

Après avoir créé une instance, vous pouvez utiliser toutes les fonctions de l'Xtra. A l'exception de AbsFileRegister(), toutes les fonctions ont besoin d'une instance de l'Xtra pour fonctionner, selon la syntaxe suivante :

<Nom de l'instance>.<Fonction>
AbsObj.OpenFile(« *.txt »)

Quand vous n'avez plus besoin de l'Xtra, libérez la mémoire qu'il occupe avec la ligne suivante :
AbsObj=VOID

AbsFileRegister(string key)

Enregistre l'Xtra. Lors de l'achat de l'Xtra, une clé de débridage vous est fournie. Cette clé doit être passée en paramètre de la fonction. Une fois l'Xtra enregistré, vous avez accès à l'ensemble des fonctions sans limitations.

Paramètre :

Key : Cette chaîne est la clé de débridage fournie lors de l'achat d'une licence de l'Xtra AbsFile.

Retourne :

Rien

Note :

Cette fonction ne nécessite pas d'instance de l'Xtra AbsFile.

Vous n'avez besoin d'enregistrer l'Xtra qu'une seule fois pour chaque session de Director. Il faut le faire avant la première utilisation d'une fonction quelconque de l'Xtra.

Si vous n'indiquez pas de numéro d'enregistrement, l'Xtra est entièrement fonctionnel. Mais une boîte d'alerte s'affichera lors de la création de chaque instance de l'Xtra.

Exemple :

AbsFileRegister(«Chaîne»)

Création d'une instance et ouverture de fichier

New()

Création d'une instance de l'Xtra. Toutes les fonctions de l'Xtra (à l'exception de AbsFileRegister) nécessitent une instance de l'Xtra pour pouvoir être exécutées.

Retourne :

Une instance de l'Xtra

Exemple :

```
gAbsFile=Xtra("AbsFile").new()
```

Open(string pathName, int openMode, int creationMode)

Ouvre un fichier.

Paramètres :

- **PathName** : C'est le chemin du fichier à ouvrir (le chemin doit être donné en absolu).

- **OpenMode** : un entier qui indique le mode d'ouverture du fichier.

0	read
1	read/write
2	shareRead
4	shareWrite

Les combinaisons sont possibles : 3 (2+1) ouvre un fichier en écriture et autorise le partage en lecture.

- **CreationMode** : c'est un entier qui indique la réaction de l'Xtra selon la présence ou non du fichier indiqué par le paramètre pathName.

0	Une erreur est déclenchée si le fichier n'existe pas.
1	Le fichier est créé si besoin
2	le fichier s'il existe est mis à zéro

Les combinaisons sont possibles. Les valeurs 1 et 2 ne sont valides que si le paramètre OpenMode est 1 ou 4 (ou toutes combinaisons comprenant l'une de ces deux valeurs).

Retourne :

- le symbol #ok si le fichier est ouvert correctement.

- le symbol #err en cas d'erreur. Dans ce cas, consultez GetError() pour obtenir le code de l'erreur.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()  
Put AbsObj.open("c:\file.dat",0,0)  
-- #ok  
AbsObj=VOID
```

Close()

Referme le dernier fichier ouvert.

Une fois un fichier ouvert avec l'appel à la fonction open, il est impossible d'ouvrir un autre fichier avec la même instance de l'Xtra. Si vous avez besoin d'ouvrir un second fichier, vous pouvez soit créer une autre instance de AbsFile, soit fermer le fichier précédemment ouvert. Pour cela, appeler la fonction close().

Si vous oubliez d'appeler cette fonction, l'appel se fera automatiquement à la destruction de l'instance.

Retourne :

- le symbol #ok si le fichier est ouvert correctement.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
-- ouverture en lecture d'un fichier existant
if AbsObj.Open("c:\file.dat", 0,0)=#err then
    put AbsObj.GetError()
end if
--read or write some data into the file
AbsObj.close()
--once the first file is close, open another file with the same instance
if AbsObj.Open("c:\anotherFile.def", 0,0)=#err then
    put AbsObj.GetError()
end if
```

OPEN/SAVE DIALOG

OpenDialog(string extension)

Affiche la boîte de dialogue d'ouverture des fichiers de Windows et ouvre le fichier en lecture.

Paramètre :

extension : indique l'extension des fichiers à afficher. Si la chaîne est vide, tous les fichiers seront visibles.

Retourne :

- une chaîne indiquant le nom du fichier et son emplacement. Celui-ci est immédiatement ouvert en lecture. Il n'est donc pas nécessaire d'appeler la fonction open().
- le symbol #cancel si on a cliqué sur annuler.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Note :

Dans la boîte de dialogue, si l'utilisateur entre au clavier un nom de fichier, l'extension passée en paramètre est automatiquement rajoutée.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.openDialog("*.dat")
-- "c:\file.dat"
AbsObj=VOID
```

SaveDialog(string extension)

Affiche la boîte de dialogue d'enregistrement de fichier de Windows et ouvre le fichier en lecture et écriture.

Paramètre :

extension : indique l'extension des fichiers à afficher. Si la chaîne est vide, tous les fichiers seront visibles.

Retourne :

- une chaîne indiquant le nom du fichier et son emplacement. Celui-ci est immédiatement ouvert en lecture. Il n'est donc pas nécessaire d'appeler la fonction open().
- le symbol #cancel si on a cliqué sur annuler.

- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Note :

Dans la boite de dialogue, si l'utilisateur entre au clavier un nom de fichier, l'extension passée en paramètre est automatiquement rajoutée.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()  
Put AbsObj.saveDialog("**.dat")  
-- "c:\file.dat"  
AbsObj=VOID
```

FileOpenDialog(string title, string filter, string defaultExt, string initDir, string defaultName, integer multiSelect, integer openFile)

Affiche une boîte de dialogue d'ouverture des fichiers.
Le ou les fichiers choisis par l'utilisateur doivent exister.

Paramètres :

title : indique le titre de la boîte de dialogue . Si la chaîne est vide, le titre par défaut de Windows est affiché.

filter :Indique le type de fichier à faire apparaître.

- Ce paramètre permet d'indiquer 1 ou plusieurs types de fichiers.

- Si la chaîne est vide, tous les fichiers apparaîtront.

- Si la chaîne n'est pas vide :

Elle doit être composée de paires de chaînes séparées par le caractère |.

« Fichier texte|*.txt » ne fera apparaître que les fichiers dont l'extension est txt. L'utilisateur verra la chaîne « Fichier texte » dans la boite de dialogue

« Fichier texte|*.txt|Fichier doc|*.doc » fera apparaître soit les fichiers dont l'extension est txt, soit ceux dont l'extension est doc. L'utilisateur devra choisir dans la liste déroulante le type de fichier à faire apparaître.

On peut également faire apparaître plusieurs types de fichiers simultanément. Pour cela, la deuxième partie d'une paire doit indiquer chaque extension, en les séparant par un ;

« Fichier texte|*.txt ;*.doc » fera apparaître simultanément les fichiers dont l'extension est txt et ceux dont l'extension est doc.

Il est possible de combiner toutes ces fonctionnalités :

« Fichier texte|*.txt ;*.doc|Fichier musique|*.wav|Fichier vidéo|*.avi ;*.mov » fera apparaître soit :

- les fichiers txt ou doc

- les fichiers wav

- les fichiers avi ou mov

defaultExt : une chaîne. indique l'extension qui sera automatiquement ajoutée au nom de fichier si l'utilisateur n'en spécifie pas. Si c'est une chaîne vide, rien ne sera rajouté.

initDir Indique le répertoire initial de la boîte de dialogue. Si la chaîne est vide, Windows gère lui-même le dernier répertoire parcouru par l'utilisateur.

DefaultName nom de fichier proposé par défaut dans le champ de saisie. La chaîne peut être vide.

multiSelect si différent de zéro, l'utilisateur peut choisir plusieurs fichiers.

openFile si différent de zéro, le fichier est automatiquement ouvert. La valeur est ignorée si l'on choisit l'option multiSelect.

Retourne :

- une chaîne : si l'utilisateur sélectionne un fichier et clique sur « ok », la chaîne retourne le chemin complet et le nom du fichier sélectionné

- une liste : si le paramètre multiSelect est différent de 0 et que l'utilisateur a sélectionné plusieurs fichiers, ceux-ci sont retournés (avec leur chemin complet) dans une liste.
- le symbol #cancel si l'utilisateur a cliqué sur le bouton « annuler »
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Note :

- 1) Si le paramètre openFile est différent de 0 et si le paramètre multiSelect est égal à 0, le fichier est automatiquement ouvert en lecture.
- 2) Pour choisir plusieurs fichiers, l'utilisateur doit les sélectionner grâce aux touches majuscule (shift) et/ou control (ctrl)

Exemple :

1) Demande de choisir un fichier quelconque. Le chemin et le nom du fichier sont retournés dans une chaîne.
 AbsObj=Xtra(« AbsFile »).new()
 Put AbsObj.fileOpenDialog("Choisissez un fichier", "", "", "", "", 0,0)
 -- "c:\file.dat"
 AbsObj=VOID

2) Demande de choisir un fichier de type "wav". Si l'utilisateur tape un nom de fichier sans extension, « .wav » sera automatiquement ajouté à la fin du nom. Le chemin et le nom du fichier sont retournés dans une chaîne. Ce fichier est automatiquement ouvert en lecture si l'utilisateur clique sur « ok ».
 AbsObj=Xtra(« AbsFile »).new()
 Put AbsObj.fileOpenDialog("Choisissez un fichier", "Fichier wav|*.wav", "wav", "", "", 0,1)
 -- "c:\file.wav"
 AbsObj=VOID

3) Demande de choisir un ou plusieurs fichiers de type txt. Si l'utilisateur choisit un seul fichier, le chemin et le nom de ce fichier sont retournés dans une chaîne. S'il choisit plusieurs fichiers simultanément, le chemin et le nom de chaque fichier sont retournés dans une liste.
 AbsObj=Xtra(« AbsFile »).new()
 Put AbsObj.fileOpenDialog("Choisissez un ou plusieurs fichiers", "Fichier texte|*.txt", "", "", "", 1,0)
 -- ["c:\aFile.txt", "c:\anotherFile.txt"]
 AbsObj=VOID

4) Demande de choisir un fichier de type texte (txt ou doc) ou musique (wav). Le chemin et le nom du fichier sont retournés dans une chaîne.
 AbsObj=Xtra(« AbsFile »).new()
 Put AbsObj.fileOpenDialog("Choisissez un fichier", "Fichier texte|*.txt ;*.doc|Fichier musique|*.wav", "", "", "", 0,1)
 -- "c:\file.doc"
 AbsObj=VOID

FileSaveDialog(string title, string filter, string defaultExt, string initDir, string defaultName, integer multiSelect, integer openFile)

Affiche une boîte de dialogue d'enregistrement de fichier.

Paramètres :

- title** : indique le titre de la boîte de dialogue . Si la chaîne est vide, le titre par défaut de Windows est affiché.
- filter** :Indique le type de fichier à faire apparaître.
- Ce paramètre permet d'indiquer 1 ou plusieurs types de fichiers.

- Si la chaîne est vide, tous les fichiers apparaîtront.
- Si la chaîne n'est pas vide :
Elle doit être composée de paires de chaînes séparées par le caractère |.
« Fichier texte|*.txt » ne fera apparaître que les fichiers dont l'extension est txt.
L'utilisateur verra la chaîne « Fichier texte » dans la boîte de dialogue
« Fichier texte|*.txt|Fichier doc|*.doc » fera apparaître soit les fichiers dont l'extension est txt, soit ceux dont l'extension est doc. L'utilisateur devra choisir dans la liste déroulante le type de fichier à faire apparaître.

On peut également faire apparaître plusieurs types de fichiers simultanément. Pour cela, la deuxième partie d'une paire doit indiquer chaque extension, en les séparant par un ;
« Fichier texte|*.txt ;*.doc » fera apparaître simultanément les fichiers dont l'extension est txt et ceux dont l'extension est doc.

Il est possible de combiner toutes ces fonctionnalités :
« Fichier texte|*.txt ;*.doc|Fichier musique|*.wav|Fichier vidéo|*.avi ;*.mov » fera apparaître soit :
- les fichiers txt ou doc
- les fichiers wav
- les fichiers avi ou mov

defaultExt : une chaîne. indique l'extension qui sera automatiquement ajoutée au nom de fichier si l'utilisateur n'en spécifie pas. Si c'est une chaîne vide, rien ne sera rajouté.

InitDir Indique le répertoire initial de la boîte de dialogue. Si la chaîne est vide, Windows gère lui-même le dernier répertoire parcouru par l'utilisateur.

DefaultName nom de fichier proposé par défaut dans le champ de saisie. La chaîne peut être vide.

multiSelect si différent de zéro, l'utilisateur peut choisir plusieurs fichiers.

openFile si différent de zéro, le fichier est automatiquement ouvert. La valeur est ignorée si l'on choisit l'option multiSelect.

Retourne :

- une chaîne : si l'utilisateur sélectionne un fichier et clique sur « ok », la chaîne retourne le chemin complet et le nom du fichier sélectionné
- une liste : si le paramètre multiSelect est différent de 0 et que l'utilisateur a sélectionné plusieurs fichiers, ceux-ci sont retournés (avec leur chemin complet) dans une liste.
- le symbol #cancel si l'utilisateur a cliqué sur le bouton « annuler »
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Note :

1) Si le paramètre openFile est différent de 0 et si le paramètre multiSelect est égal à 0, le fichier est automatiquement ouvert en lecture.

2) Pour choisir plusieurs fichiers, l'utilisateur doit les sélectionner grâce aux touches majuscule (shift) et/ou control (ctrl)

Exemple :

1) Demande de choisir un fichier quelconque. Le chemin et le nom du fichier sont retournés dans une chaîne.

```
AbsObj=Xtra(« AbsFile »).new()  
Put AbsObj.fileOpenDialog("Choisissez un fichier", "", "", "", "", 0,0)  
-- "c:\file.dat"  
AbsObj=VOID
```

2) Demande de choisir un fichier de type "wav". Si l'utilisateur tape un nom de fichier sans extension, « .wav » sera automatiquement ajouté à la fin du nom. Le chemin et le nom du fichier sont retournés dans une chaîne. Ce fichier est automatiquement ouvert en lecture si l'utilisateur clique sur « ok ».

```
AbsObj=Xtra(« AbsFile »).new()  
Put AbsObj.fileOpenDialog("Choisissez un fichier", "Fichier wav|*.wav", "wav", "", "", 0,1)
```

```
-- "c:\file.wav"  
AbsObj=VOID
```

3) Demande de choisir un ou plusieurs fichiers de type txt. Si l'utilisateur choisit un seul fichier, le chemin et le nom de ce fichier sont retournés dans une chaîne. S'il choisit plusieurs fichiers simultanément, le chemin et le nom de chaque fichier sont retournés dans une liste.

```
AbsObj=Xtra(« AbsFile »).new()  
Put AbsObj.fileSaveDialog("Choisissez un ou plusieurs fichiers", "Fichier texte|*.txt", "",  
"", "", 1,0)  
-- ["c:\aFile.txt", "c:\anotherFile.txt"]  
AbsObj=VOID
```

4) Demande de choisir un fichier de type texte (txt ou doc) ou musique (wav). Le chemin et le nom du fichier sont retournés dans une chaîne.

```
AbsObj=Xtra(« AbsFile »).new()  
Put AbsObj.fileSaveDialog ("Choisissez un fichier", "Fichier texte|*.txt ;*.doc|Fichier  
musique|*.wav", "", "", "", 0,1)  
-- "c:\file.doc"  
AbsObj=VOID
```

FolderDialog(string title)

Affiche la boîte de dialogue de choix d'un dossier de Windows.

Paramètre :

Title : indique le titre de la boîte de dialogue.

Retourne :

- une chaîne : le chemin et le nom du dossier sélectionné.
- le symbol #cancel si l'utilisateur a cliqué sur « annuler ».
- le symbol #err en cas d'erreur. Consultez GetLastError() pour obtenir le code de l'erreur.

Note :

- 1) Seuls les dossiers valides peuvent être sélectionnés : « Poste de travail », « voisinage réseau », ... ne peuvent pas être sélectionnés.
- 2) Le chemin se termine toujours par un \

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()  
Put AbsObj.folderDialog("Choisissez un dossier")  
-- "c:\program files\  
AbsObj=VOID
```

ComputerDialog(string title)

Affiche une boîte de dialogue pour sélectionner un ordinateur.

Paramètre :

Title : indique le titre de la boîte de dialogue.

Retourne :

- une chaîne : le nom de l'ordinateur sélectionné.
- le symbol #cancel si l'utilisateur a cliqué sur « annuler ».
- le symbol #err en cas d'erreur. Consultez GetLastError() pour obtenir le code de l'erreur.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()  
Put AbsObj.computerDialog("Choisissez un ordinateur")  
-- "\\computer"  
AbsObj=VOID
```

PrinterDialog(string title)

Affiche une boîte de dialogue pour sélectionner une imprimante. L'utilisateur peut sélectionner une imprimante reliée à un ordinateur local ou au réseau local.

Paramètre :

Title : le nom de la boîte de dialogue

Retourne :

- une chaîne : le chemin et le nom de l'imprimante sélectionnée.
- le symbol #cancel si l'utilisateur a cliqué sur « annuler ».
- le symbol #err en cas d'erreur. Consultez GetLastError() pour obtenir le code de l'erreur.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.computerDialog("Choisissez un ordinateur")
-- "\\computer\sharedprinter"
AbsObj=VOID
```

Fonctions de paramétrage

SetIntegerFormat(integer zeroForWindows)

Fixe le format de lecture et d'écriture des entiers.

SetIntegerFormat(0) (format par défaut) pour le format little endian (PC Windows).

SetIntegerFormat(1) pour le format big endian (autres plates-formes).

La fonction concerne les entiers courts (short : 16 bits) et long (int : 32 bits), signés ou non.

L'appel est nécessaire pour lire/écrire certains fichiers qui ne sont pas propres au monde du PC. Ex : fichiers MP3, TIFF etc.

Paramètre :

ZeroForWindows :
0 – lit/écrit au format little endian
1 – lit/écrit au format big endian

Retourne :

rien

Note :

- 1) Par défaut, toutes les fonctions de lecture ou d'écriture sont au format little endian.
- 2) Vous devez déterminer le bon format à utiliser. Rien ne permet de savoir à l'avance le format utilisé.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.setIntegerFormat(0)
-- "\\computer\sharedprinter"
AbsObj=VOID
```

Fonctions de lecture

Les valeurs sont lues à partir de la position actuelle du pointeur de fichier. Le fichier est donc lu séquentiellement.

Ces fonctions retournent #err en cas d'erreur ou #eof si la tête de lecture se retrouve en dehors du fichier (c'est-à-dire après la fin du fichier).

ReadByte()

lit un octet signé et avance le pointeur d'un octet.

retourne :

- un entier signé. (-128 → 127)
- le symbol #eof si le pointeur du fichier est situé après la fin du fichier.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",0,0)
-- #Ok
Put AbsObj.readByte()
-- 50
AbsObj=VOID
```

ReadUByte()

lit un octet et avance le pointeur d'un octet.

retourne :

- un entier non signé. (0 → 255)
- le symbol #eof si le pointeur du fichier est situé après la fin du fichier.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",0,0)
-- #Ok
Put AbsObj.readUByte()
-- -50
AbsObj=VOID
```

ReadShort()

lit un entier 16 bits et avance le pointeur de 2 octets.

retourne :

- un entier 16 bits signé. (-32768 → 32767)
- le symbol #eof si le pointeur du fichier est situé après la fin du fichier.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",0,0)
-- #Ok
Put AbsObj.readShort()
-- 4830
AbsObj=VOID
```

ReadUShort()

lit un entier 16 bits et avance le pointeur de 2 octets.

retourne :

- un entier 16 bits non signé. (0 → 65535)

- le symbol #eof si le pointeur du fichier est situé après la fin du fichier.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",0,0)
-- #Ok
Put AbsObj.readUShort()
-- -7520
AbsObj=VOID
```

ReadInt()

lit un entier 32 bits et avance le pointeur de 4 octets.

retourne :

- un entier 32 bits signé. (-2147483648 → 2147483647)
- le symbol #eof si le pointeur du fichier est situé après la fin du fichier.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",0,0)
-- #Ok
Put AbsObj.readInt()
-- 16485440
AbsObj=VOID
```

ReadUInt()

lit un entier 32 bits et avance le pointeur de 4 octets.

Retourne :

- une valeur non signée (0 → 4294967295)
- un entier (de 0 à 2147483647)
- un float (de 2147483648 à 4294967295)
Ceci est dû au fait que les entiers Director ne peuvent dépasser 2147483647 ($2^{31} - 1$)
- le symbol #eof si le pointeur du fichier est situé après la fin du fichier.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",0,0)
-- #Ok
Put AbsObj.readUInt()
-- -84659660
AbsObj=VOID
```

ReadFloat()

lit un nombre décimal signé IEEE 64 bits (format interne des décimaux Director) et avance le pointeur de 8 octets

Retourne

- un float (+/- 2.2250738585072014 E-308 → +/- 1.7976931348623158 E+308)
- le symbol #eof si le pointeur du fichier est situé après la fin du fichier.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

exemple :

```
AbsObj=Xtra(« AbsFile »).new()
```

```
Put AbsObj.open("c:\file.dat",0,0)
-- #Ok
Put AbsObj.readFloat()
-- 1.00059636159647e33
AbsObj=VOID
```

ReadRect()

Lit un rectangle au format Director et avance le pointeur de fichier de 16 octets

Retourne :

- un rect Director (rect(top,left,bottom,right))
- le symbol #eof si le pointeur du fichier est situé après la fin du fichier.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",0,0)
-- #Ok
Put AbsObj.readRect()
-- rect(10,50,100,200)
AbsObj=VOID
```

ReadPoint()

Lit un point au format Director et avance le pointeur de fichier de 8 octets.

Retourne :

- un point Director (point(y,x))
- le symbol #eof si le pointeur du fichier est situé après la fin du fichier.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",0,0)
-- #Ok
Put AbsObj.readPoint()
-- point(50,100)
AbsObj=VOID
```

ReadSymbol()

lit un symbol (voir la fonction WriteSymbol). Le pointeur de fichier avance d'un nombre d'octets égal à 2 + la longueur du texte du symbol (#symbol avance de 8 octets = 2 + 6)

Retourne :

- un symbol
- le symbol #eof si le pointeur du fichier est situé après la fin du fichier.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",0,0)
-- #Ok
Put AbsObj.readSymbol()
-- #text
AbsObj=VOID
```

ReadBinary(integer nBytes)

Lit une suite d'octet et les place dans une chaîne de caractères. Le pointeur est déplacé de nBytes

Paramètre :

nBytes : le nombre de caractères à lire
Si nBytes<0, alors, lit le fichier jusqu'à la fin

Retourne :

- une chaîne. Un zéro est ajouté en fin de chaîne.
- le symbol #eof si le pointeur du fichier est situé après la fin du fichier.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Note :

La fonction readBinary interprète la suite d'octets lus comme une chaîne de caractères.

Pour Director une chaîne de caractères se termine toujours par un 0. Ainsi si un 0 se trouve au milieu des octets lus, Director considèrera que la chaîne se termine à ce caractère. Mais la chaîne en mémoire reste bien celle qui est lue.

Exemple avec la chaîne « Hello world » :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",0,0)
-- #Ok
Put AbsObj.writestring(« Hello »,#nil)
-- 5
put AbsObj.writeByte(0)
-- 1
put AbsObj.write("world",#nil)
-- 5
AbsObj.setPosition(0,0)
Str=AbsObj.readBinary(11)
Put Str
-- "Hello"
Put Str.length
-- 11
Put Str.char[10]
-- "l"
```

L'affichage de la chaîne complète s'arrête donc au 5^{ème} caractère, le 6^{ème} étant un 0. Mais la longueur reste 11, et le 10ème caractère est bien le l de world.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",0,0)
-- #Ok
Put AbsObj.readBinary(5)
-- "Hello world"
AbsObj=VOID
```

ReadString(integer maxChar)

Lit une chaîne de maxChar caractères au maximum. Avance le pointeur de fichier du nombre d'octets réellement lus.

Paramètre :

MaxChar : le nombre maximum de caractères à lire.
Si maxChar < 0, lit le fichier de sa position actuelle jusqu'à la fin ou jusqu'au prochain caractère 0, CR (carriage return) ou LF (line Feed), CR + LF et Ctrl+Z (fin de texte).
La lecture s'arrête au 1^{er} 0, CR, LF ou Ctrl+Z rencontré, que maxChar soit < 0 ou non

Retourne :

- une chaîne : la chaîne lue. Un zéro est toujours ajouté en fin de chaîne.
- le symbol #eof si le pointeur du fichier est situé après la fin du fichier.

- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Note :

Quel que soit le nombre de caractère demandé (maxChar), la lecture s'arrête soit au 1^{er} zéro, CR, LF (retour à la ligne)
- soit à l'indicateur Ctrl+Z
- soit si la fin du fichier est atteinte.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()  
Put AbsObj.open("c:\file.dat",0,0)  
-- #Ok  
Put AbsObj.readString(11)  
-- "Hello World"  
AbsObj=VOID
```

ReadText(integer maxChar)

Lit une chaîne de maxChar caractères au maximum.

Lit une chaîne de caractères. La lecture s'arrête au 1^{er} zéro ou Ctrl+Z (fin de texte), Fin de fichier. La différence avec readString est que cette fonction lit les CR et LF.

Avance le pointeur de fichier du nombre d'octets réellement lus.

Paramètre :

MaxChar : le nombre maximum de caractères à lire
Si maxChar < 0, lit le fichier de sa position actuelle jusqu'à la fin ou jusqu'au prochain caractère 0.

Retourne :

- une chaîne : la chaîne lue. Un zéro est toujours ajouté en fin de chaîne.
- le symbol #eof si le pointeur du fichier est situé après la fin du fichier.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Note :

Quel que soit le nombre de caractères demandé (maxChar), la lecture s'arrête au 1^{er} zéro rencontré ou si la fin du fichier est atteinte.
La différence avec ReadString est que cette fonction lit les CR et les LF.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()  
Put AbsObj.open("c:\file.dat",0,0)  
-- #Ok  
Put AbsObj.readText(11)  
-- "Hello World"  
AbsObj=VOID
```

ReadList(string listName)

Lit une liste enregistrée avec WriteList sous le nom listName.

Le fichier n'a pas besoin d'être positionné au début de la liste : la fonction retrouve automatiquement la liste par son nom.

La fonction est capable de lire les listes imbriquées, ainsi que les listes de propriétés.

Paramètre :

ListName : le nom d'une liste préalablement écrite avec writeList(). Attention : le nom est sensible à la casse : « aName » est différent de « aname »

Retourne :

- une liste. Il s'agit de la liste précédemment écrite.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

BytesReaded()

Indique le nombre d'octets lus par la dernière fonction de lecture appelée.

Retourne :

- un entier : le nombre d'octets lus si celui-ci est inférieur ou égal à 2147483647
- un float : le nombre d'octets lus si celui-ci est supérieur à 2147483647

exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",0,0)
-- #Ok
Put AbsObj.readInt()
-- 59
Put AbsObj.bytesReaded()
-- 4
AbsObj=VOID
```

Fonctions d'écriture

WriteByte(integer value)

Écrit un octet signé et avance le pointeur de 1 octet. Si le fichier est trop petit pour contenir la valeur, ou si le pointeur de fichier est situé à la fin du fichier, le fichier est agrandi.

Paramètre :

Value : un entier de -128 à +127. Si value ne peut être contenu dans un octet (inférieur à -128 ou supérieur à 127, seul l'octet de poids faible est écrit)

Retourne :

- un entier : la fonction a bien fonctionné. L'entier indique le nombre d'octets écrits.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.writeByte(-10)
-- 1
AbsObj=VOID
```

WriteUByte(integer value)

Écrit un octet non signé et avance le pointeur de 1 octet. Si le fichier est trop petit pour contenir la valeur, ou si le pointeur de fichier est situé à la fin du fichier, le fichier est agrandi.

Paramètre :

Value : un entier de 0 à 255

Retourne :

- un entier : la fonction a bien fonctionné. L'entier indique le nombre d'octets écrits.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",1,3)
```

```
-- #Ok
Put AbsObj.writeUByte(10)
-- 1
AbsObj=VOID
```

WriteShort(integer value)

Écrit un entier 16 bit signé et avance le pointeur de 2 octets. Si le fichier est trop petit pour contenir la valeur, ou si le pointeur de fichier est situé à la fin du fichier, le fichier est agrandi.

Paramètre :

Value : un entier de -32768 à +32767

Retourne :

- un entier : la fonction a bien fonctionné. L'entier indique le nombre d'octets écrits.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.writeShort(-4580)
-- 2
AbsObj=VOID
```

WriteUShort (integer value)

Écrit un entier 16 bit signé et avance le pointeur de 2 octets. Si le fichier est trop petit pour contenir la valeur, ou si le pointeur de fichier est situé à la fin du fichier, le fichier est agrandi.

Paramètre :

Value : un entier de 0 à 65535

Retourne :

- un entier : la fonction a bien fonctionné. L'entier indique le nombre d'octets écrits.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.writeUShort(4620)
-- 2
AbsObj=VOID
```

WriteInt(integer value)

Écrit un entier 32 bit signé et avance le pointeur de 4 octets. Si le fichier est trop petit pour contenir la valeur, ou si le pointeur de fichier est situé à la fin du fichier, le fichier est agrandi.

Paramètre :

Value : un entier de -2147483648 à 2147483647

Retourne :

- un entier : la fonction a bien fonctionné. L'entier indique le nombre d'octets écrits.

- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.writeInt(-16777650)
-- 4
AbsObj=VOID
```

WriteUInt(float value)

Écrit un entier 32 bit non signé et avance le pointeur de 4 octets. Si le fichier est trop petit pour contenir la valeur, ou si le pointeur de fichier est situé à la fin du fichier, le fichier est agrandi.

Paramètre :

Value : un entier de 0 à 4294967295

Retourne :

- un entier : la fonction a bien fonctionné. L'entier indique le nombre d'octets écrits.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.writeUInt(2789684520)
-- 4
AbsObj=VOID
```

WriteFloat(float value)

écrit un nombre décimal signé IEEE 64 bits (format interne des décimaux Director) et avance le pointeur de 8 octets Si le fichier est trop petit pour contenir la valeur, ou si le pointeur de fichier est situé à la fin du fichier, le fichier est agrandi.

Paramètre :

Value : un nombre flottant

Retourne

- un entier : la fonction a bien fonctionné. L'entier indique le nombre d'octets écrits.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.writeFloat(1536.265542)
-- 8
AbsObj=VOID
```

WriteSymbol(symbol value)

Écrit un symbol : dans le fichier le symbol est stocké sous la forme d'une chaîne précédée de # et terminée par 0. Si le fichier est trop petit pour contenir la valeur, ou si le pointeur de fichier est situé à la fin du fichier, le fichier est agrandi.
Le pointeur de fichier avance d'un nombre d'octet égale à 2+la longueur du texte du symbol

Paramètre :

Value : un symbol Director

Retourne :

- un entier : la fonction a bien fonctionné. L'entier indique le nombre d'octets écrits.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.writeSymbol(#symbol)
-- 8
AbsObj=VOID
```

Write(any)

Écrit une ou plusieurs valeurs Director. Si le fichier est trop petit pour contenir la valeur, ou si le pointeur de fichier est situé à la fin du fichier, le fichier est agrandi. Le pointeur de fichier avance d'un nombre d'octets dépendant de la ou des variables écrites.

Paramètre :

Any : 1 ou plusieurs paramètres Director, séparés par des virgules.
Les types acceptés sont integer, float, string, symbol, point, rect, list et proplist.
Les éléments des listes doivent être également des integer, float, string, symbol, point, rect, list et proplist.

Retourne :

- un entier : la fonction a bien fonctionné. L'entier indique le nombre d'octets écrits.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Note :

Les chaînes sont écrites avec un zéro final.

Exemple :

Utilisation simple de la fonction write()

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.write(55)
-- 4
Put AbsObj.write("Hello")
-- 6
AbsObj=VOID
```

Utilisation de plusieurs paramètres

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.write(55,"Hello",point(10,20))
-- 18
AbsObj=VOID
```

WriteString(string aString, symbol terminator)

Écrit une chaîne en permettant de définir le caractère de fin. Si le fichier est trop petit pour contenir la valeur, ou si le pointeur de fichier est situé à la fin du fichier, le fichier est agrandi.

Paramètres :

aString : la chaîne de caractère à écrire.
Terminator : un symbol.
#zero ou **#o** un zéro est stocké en fin de chaîne
(identique à Write(string))
#crlf un retour à la ligne (format PC) est ajouté
#cr un retour chariot seul
#end ou **#z** une marque de fin de texte est ajoutée
#- ou **#nil** rien du tout

Retourne :

- un entier : la fonction a bien fonctionné. L'entier indique le nombre d'octets écrits.
- le symbol **#err** en cas d'erreur. Consultez **GetError()** pour obtenir le code de l'erreur.

exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.writeString("Hello World",#zero)
-- 12
AbsObj=VOID
```

WriteList(string listName, list theList)

Ecrit une liste (liste simple ou proplist).
Tous les types simples (voir Write) sont acceptés pour les éléments. (integer, float, string, symbol, point, rect, list et void).
Une liste peut contenir d'autres listes qui peuvent elles-mêmes en contenir...

Paramètres :

ListName : un nom pour identifier la liste écrite. Ce nom est nécessaire pour relire la liste grâce à **readList()**. Attention : le nom est sensible à la casse : « aName » est différent de « aname »
TheList : la liste à écrire. Cette liste accepte tous les types simples de Director. Il peut s'agir d'une liste linéaire ou de propriété. La liste peut contenir des listes imbriquées

Retourne :

- un entier : la fonction a bien fonctionné. L'entier indique le nombre d'octets écrits.
- le symbol **#err** en cas d'erreur. Consultez **GetError()** pour obtenir le code de l'erreur.

Note :

Un fichier ne peut contenir qu'une seule liste pour un même nom. Si vous essayez d'écrire une liste en donnant comme nom celui d'une autre liste préalablement écrite dans le fichier, cette dernière liste sera remplacée par la nouvelle. Ainsi :

```
AbsObj.writeList("aName",[1,2,3])
Put AbsObj.readList("aName")
-- [1, 2, 3]
AbsObj.writeList(,aName",[4,5,6])
Put AbsObj.readList("aName")
-- [4, 5, 6]
```

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.writeList( "ListName", [#name:"Andrew", #firstName:"Paul", #age:42,
#children:[[#firstName:"Julia", #age:12], [#firstName:"Joe", #age:15]]])
-- 193
AbsObj=VOID
```

Fonctions de pointeur et taille de fichier

Les fonctions renvoient #err en cas d'erreur.

Length()

Renvoie la taille du fichier en nombre d'octets.

Retourne :

- un entier si la taille est inférieure ou égale à 2147483647
- un float si la taille est supérieure à 2147483647
- le symbol #err en cas d'erreur. Consultez GetLastError() pour obtenir le code de l'erreur.

Exemple :

```
AbsObj=Xtra(« AbsObj »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.length()
-- 178250
AbsObj=VOID
```

GetPosition()

Retourne la position actuelle du pointeur de fichier.

Retourne :

- un entier indiquant la position du pointeur.
- le symbol #err en cas d'erreur. Consultez GetLastError() pour obtenir le code de l'erreur.

Note :

GetPosition ne retourne pas la position correcte si la taille du fichier est supérieure à 4 Go. La fonction retourne une valeur entière, mais qui peut être erronée.

Exemple :

```
AbsObj=Xtra(« AbsObj »).new()
Put AbsObj.open("c:\file.dat",0,0)
-- #Ok
Put AbsObj.readByte()
-- 45
Put AbsObj.getPosition()
-- 4
AbsObj=VOID
```

SetPosition(integer position, integer mode)

Positionne le pointeur de fichier.

Le pointeur peut se trouver après la fin du fichier sans causer d'erreur. Si le fichier est ouvert en lecture, un #eof sera renvoyé lors de la prochaine utilisation d'une fonction de lecture. Si le fichier est ouvert en écriture, il sera agrandi jusqu'à cette position lors de la prochaine opération d'écriture ou lors de l'appel à la fonction setEndOfFile().

Paramètre :

Position : un entier. Indique le nombre d'octets pour le déplacement.

Mode : un entier.

0 (FILE_BEGIN), le déplacement du pointeur est calculé par rapport au début du fichier. Position est forcément >=0

1 (FILE_CURRENT), le déplacement du pointeur est calculé par rapport à la position actuelle (position est positif ou négatif).

2 (FILE_END), le déplacement est calculé par rapport à la fin du fichier. Position peut être positif ou négatif.

Retourne :

- un entier : la nouvelle position absolue sur pointeur.
- le symbol #err en cas d'erreur. Consultez GetLastError() pour obtenir le code de l'erreur.

Note :

Il n'est pas possible d'adresser une position absolue supérieure à 2 Go. Ainsi setPosition(2147483648,0) ne fonctionnera pas correctement (2147483648>2Go). Il est par contre possible de positionner le pointeur de fichier à une position supérieure à 2 Go, par déplacement relatif.

 setPosition(1073741824,0) –positionne le pointeur sur 1 Go

 setPosition(1073741824,1) –positionne le pointeur 1 Go plus loin

 setPosition(1073741824,1) –positionne le pointeur encore 1 Go plus loin. Le pointeur de fichier se trouve maintenant à 3 Go du début du fichier.

Exemple :

```
AbsObj=Xtra(« AbsObj »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
-- Positionne le pointeur de fichier 20 octets après le début du fichier
Put AbsObj.setPosition(20,0)
-- 20
-- Déplace le pointeur de fichier de 30 octets à partir de sa position actuelle
Put AbsObj.setPosition(30,1)
-- 30
AbsObj=VOID
```

SetEndOfFile()

Fixe la fin du fichier à la position actuelle du pointeur. Le fichier est tronqué si le pointeur est positionné avant la fin du fichier ou agrandi s'il est positionné après.

Retourne :

- un entier si la nouvelle taille du fichier est inférieure ou égale à 2147483647
- un float si la nouvelle taille est supérieure à 2147483648
- le symbol #err en cas d'erreur. Consultez GetLastError() pour obtenir le code de l'erreur.

Note :

Windows 95 et 98 ne supportent pas les fichiers de plus de 2 Go

Exemple :

```
AbsObj=Xtra(« AbsObj »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
AbsObj.SetPosition(0,0)
AbsObj.SetEndOfFile() -- le fichier est remis à zéro
AbsObj.SetPosition(1000, 0)
size = myFile.SetEndOfFile() -- le fichier est agrandi de 1000 octets
AbsObj=VOID
```

Fonctions générales

SetMarker(string name)

Écrit un marqueur dans le fichier. Ce marqueur peut être utilisé pour localiser facilement une valeur. Déplace le pointeur de fichier de (4+longueur du nom) octets. Ces marqueurs permettent de définir une position dans le fichier, et de pouvoir facilement y retourner grâce à la fonction gotoMarker().

Paramètre :

Name : une chaîne. La chaîne qui va servir de marqueur.

Retourne :

- un entier : le nombre d'octets écrits.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Note :

Le nom passé en paramètre est sensible à la casse. « aName » est différent de « aname ».

Exemple :

```
AbsObj=Xtra(« AbsObj »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.setMarker("aName")
-- 9
AbsObj=VOID
```

GotoMarker(string name)

Déplace le pointeur de fichier juste après le marqueur écrit avec SetMarker.

Paramètre :

Name : le nom du marqueur à rejoindre.

Retourne :

- #ok : le marqueur a été trouvé, et le pointeur est correctement positionné.
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Note :

Le nom passé en paramètre est sensible à la casse. « aName » est différent de « aname ».

Exemple :

```
AbsObj=Xtra(« AbsObj »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.gotoMarker("aName")
-- #Ok
AbsObj=VOID
```

Flush()

Force l'écriture de tous les caches sur disque. Windows peut différer l'écriture de données sur support physique. Cette fonction permet de forcer l'écriture réelle des données.

Retourne :

- le symbol #ok : les caches ont été correctement écrits
- le symbol #err en cas d'erreur. Consultez GetError() pour obtenir le code de l'erreur.

Exemple :

```
AbsObj=Xtra(« AbsObj »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.writeInt(133)
-- 4
Put AbsObj.flush()
-- #Ok
AbsObj=VOID
```

Fonctions d'erreur

GetError()

Renvoie le code de la dernière erreur générée. Ce code correspond aux codes d'erreur de Windows (0 = pas d'erreur).

Retourne :

Un entier correspondant à la dernière erreur générée par un appel à l'une des fonctions de l'Xtra Absfile. Toutes les fonctions modifient ce code lors de leur exécution (à l'exception de GetError()) même lorsque la fonction s'est correctement terminée. Dans ce cas, la valeur 0 est retournée.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.open("c:\file.dat",1,3)
-- #Ok
Put AbsObj.getError()
-- 0
AbsObj=VOID
```

GetErrorText(integer codeErreur)

Renvoie le texte de description correspondant à l'erreur codeErreur.

Paramètre :

CodeErreur : un entier correspondant à un code d'erreur Windows.

Retourne :

Une chaîne correspondant au texte de l'erreur passée en paramètre. Cette chaîne est dépendante du système d'exploitation (version et langue).

Note :

Les messages d'erreurs sont dépendants du système d'exploitation de l'utilisateur, ainsi que de la langue utilisée.

Exemple :

```
AbsObj=Xtra(« AbsFile »).new()
Put AbsObj.getErrorText(2)
-- « Le fichier spécifié est introuvable. »
AbsObj=VOID
```

Codes d'erreur

Le programme ErrorInfo.exe est fourni avec l'Xtra. Il vous permet de déterminer le texte d'erreur pour un code donné.

- Lancez ErrorInfo.exe
- Tapez le numéro d'une erreur.
- Cliquez sur « Find »

Codes d'erreurs les plus fréquents :

1	ERROR_INVALID_FUNCTION	Appel d'une fonction alors qu'aucun fichier n'est ouvert
2	ERROR_FILE_NOT_FOUND	Fichier introuvable
3	ERROR_PATH_NOT_FOUND	Chemin d'accès introuvable
5	ERROR_ACCESS_DENIED	Accès refusé : par exemple tentative d'écriture dans un fichier ouvert en lecture seule ou sur un CD-Rom, tentative d'accès à un fichier alors que l'on ne possède pas les autorisations nécessaires, fichier crypté etc.
8	ERROR_NOT_ENOUGH_MEMORY	Pas assez de mémoire. Peut survenir lors de la lecture de données gigantesques.
23	ERROR_CRC	Erreur de lecture/écriture de données due au support.
32	ERROR_SHARING_VIOLATION	Erreur de partage
87	ERROR_INVALID_PARAMETER	Mauvais paramètre : retourné par Write ou WriteList s'ils contiennent des types de données non gérés par la fonction.
2012	ERROR_TAG_NOT_FOUND	Le marqueur ou le nom de la liste n'a pas été trouvé.