Guide AppleScript des Compléments de Pilotage Standards

version française

Préambule

Ce guide n'est absolument pas une traduction officielle de la Société Apple.

Ce guide est basé sur le guide *AppleScript Scripting Additions Guide* uniquement disponible en version anglaise depuis 1997. À ce jour, aucune localisation de ce texte n'a été faite, que ce soit en français ou dans une autre langue.

Ce guide est la suite logique de la précédente traduction : *Guide AppleScript version française* disponible sur le site <trad.applescript.free.fr/Accueil.html>.

Ce guide n'est pas exempt d'erreurs de frappe ou d'incohérences dans la rédaction et je vous prie de m'en excuser.

Dans l'espoir que cette version française comblera l'attente de tous les utilisateurs francophones, je vous souhaite une bonne lecture des commandes fournies avec AppleScript.

Un utilisateur Mac francophone

Merci à Daniel, Pierre, Raymond et Denis pour leur aide et leurs conseils indispensables.

Sommaire

AppleScript est un langage de pilotage, d'automatisation de tâches, que ce soit pour le système ou pour des applications. Au lieu d'utiliser la souris ou le clavier pour manipuler les menus ou les boutons, vous pouvez écrire un jeu d'instructions - appelé script - pour automatiser les tâches répétitives ou pour personnaliser les applications.

Tous les caractères de couleur bleue sont des liens qui afficheront la page indiquée.

Ce guide comporte les chapitres suivants :

* Commande rajoutée par rapport au guide Apple

* Commande rajoutée par rapport au guide Apple

Conventions

Certains termes de ce guide sont mis en **gras** lorsqu'ils sont définis pour la première fois.

Les conventions suivantes sont utilisées dans la syntaxe de ce guide :

`élément de langage`	indique un exemple ou morceau de script. S'il y a un symbole (par exemple, + ou &), celui-ci fait partie du texte du script et doit être saisi dans la fenêtre de saisie de l'Éditeur de scripts.
paramètre de substitution	un texte en italique indique un *paramètre de substitution* que vous remplacerez par une valeur appropriée. (Dans d'autres langages de programmation, les *paramètres de substitution* sont appelés nonterminals.)
[optionnel]	les crochets indiquent que les éléments inscrits entre crochets sont facultatifs lors de l'écriture des scripts.
(un groupe)	les parenthèses groupent ensemble plusieurs éléments. Si les parenthèses font partie de la syntaxe du langage, elles sont **en gras**.
[optionnel]...	trois petits points (...) après un groupe défini entre crochets indiquent que vous pouvez répéter le groupe d'éléments entre crochets x fois (1 ou plus).
(un groupe)...	trois petits points (...) après un groupe défini entre parenthèses indiquent que vous pouvez répéter le groupe d'éléments entre parenthèses x fois (1 ou plus).
a | b | c	les barres verticales séparent les éléments d'un groupe dans lequel vous devez choisir un seul élément. Les éléments séparés par des barres verticales sont le plus souvent groupés entre crochets ou parenthèses.

Chapitre 2

Préface

Ce guide décrit la plupart des commandes fournies en standard, en tant que complément de pilotage, avec AppleScript. Les compléments de pilotage sont des fichiers qui étendent les capacités du langage AppleScript, en fournissant des commandes supplémentaires utilisables dans les scripts.

Avant d'utiliser ce guide, il est chaudement recommandé de parcourir l'aide AppleScript du Centre d'aide Mac OS et, de lire le *Guide AppleScript version française* (trad.applescript.free.fr/Accueil.html)afin de connaître les bases du langage.

La majorité des commandes détaillées dans le chapitre "Les définitions de commandes" sont tirées du guide *AppleScript Scripting Additions Guide version anglaise* disponible sur le site d'Apple <www.apple.com/applescript/>.

De plus, par rapport à la version anglaise, certaines définitions de commandes ont été rajoutées, mais pas toutes, comme Choose From List ou Delay.

Par contre, les définitions des commandes n'étant pas définitives, l'implémentation des commandes peut varier. Ce guide a été traduit en fonction de ma configuration matérielle, OS 8.6 AS 1.3.7 Compléments Standard 1.3.7. Par conséquent, si votre configuration matérielle diffère, je vous conseille de vous référer à la documentation fournie dans l'aide AppleScript, aux dictionnaires de ces commandes, ou si la langue anglaise ne vous rebute pas , au site de l'AppleScript Sourcebook : <http://www.AppleScriptSourcebook.com/applescript.html>.

Chapitre 3

Introduction aux compléments de pilotage

Les compléments de pilotage sont des fichiers, qui étendent les capacités du langage AppleScript, en fournissant des commandes supplémentaires ou des coercitions utilisables dans les scripts. Ils sont similaires aux XCMDs et XFCNs dans HyperTalk.

Ce chapitre indique comment installer les compléments de pilotage, et décrit brièvement le fonctionnement de ceux-ci. Pour plus d'informations sur les différences entre les commandes des compléments de pilotage, les commandes d'application, les commandes AppleScript et les commandes définies par l'utilisateur, voir *Guide AppleScript version française* tome 2 "Les types de commandes".

Installer les compléments de pilotage

Lors de l'installation d'AppleScript, installation qui se fait en même temps que Mac OS, l'installateur crée un dossier "Compléments de pilotage" dans le dossier Système. Il y copie plusieurs compléments de pilotage dont "Compléments standard".

➥ **Note des traducteurs francophones**
Avant Mac OS 8.0 (AS 1.1.2), le dossier "Compléments de pilotage" était installé dans le dossier "Extensions". De même, l'OSAX "Compléments standard" est apparu avec Mac OS 8.5 (AS 1.3.4), avant chaque commande avait son fichier propre, maintenant ces commandes sont regroupées dans un seul fichier. ●

Chaque complément de pilotage contient un ou plusieurs gestionnaires de commandes. Si un complément de pilotage est installé dans le dossier "Compléments de pilotage", les gestionnaires de commandes qu'il contient seront utilisables par n'importe quel script, dont la cible est une application sur cet ordinateur.

Certains compléments de pilotage définissent aussi des classes d'objet pour les enregistrements retournés par leurs commandes. Toutefois, normalement, les commandes des compléments de pilotage n'agissent pas sur les objets

définis par les applications.

Pour qu'un complément de pilotage, fourni par Apple ou par une tierce partie, soit reconnu par AppleScript, il doit être obligatoirement présent dans le dossier "Compléments de pilotage".

Si dans un script, lors de l'utilisation d'une commande d'un complément de pilotage, vous obtenez un message d'erreur indiquant que la commande n'est pas définie, vérifiez que le complément de pilotage correspondant est bien installé dans le dossier "Compléments de pilotage".

Envoyer des commandes de compléments de pilotage

Comme la cible d'une commande d'application, la cible d'une commande de complément de pilotage est toujours un objet d'application ou un script-objet. Si le script ne spécifie pas explicitement la cible avec une instruction Tell, AppleScript envoie la commande à l'application cible par défaut, généralement l'application qui exécute le script (par exemple, l'Éditeur de scripts).

Une commande de complément de pilotage n'exécute ses actions qu'après que l'application cible ait reçu la commande. À la différence des commandes d'application, les commandes des compléments de pilotage fonctionnent toujours de la même façon, quelle que soit l'application qui les reçoit.

Par exemple, la commande Display Dialog affiche une boîte de dialogue pouvant comporter du texte, un ou plusieurs boutons, un icone et un champ dans lequel l'utilisateur peut saisir des caractères. Dans le script qui suit, la cible de la commande Display Dialog est l'application AppleWorks. Lorsque le script tourne, l'application AppleWorks devient l'application en avant-plan (c'est à dire que ses menus deviennent visibles et ses fenêtres sont mises au premier plan dans l'écran), elle transmet alors la commande au gestionnaire de la commande Display Dialog, et ce gestionnaire affiche la boîte de dialogue.

```
tell application "AppleWorks"
   display dialog "Quel est ton nom ? " default answer ""
end tell
```

Dans l'exemple suivant, la commande Display Dialog n'est pas encadrée par une instruction Tell, et n'a pas non plus de paramètre direct, aussi sa cible est

l'Éditeur de scripts (ou une autre application exécutant le script). Lorsque vous lancez le script, l'Éditeur de scripts transmet la commande au gestionnaire de la commande Display Dialog, lequel affiche la boîte de dialogue au niveau de l'Éditeur de scripts (c'est à dire, par-dessus n'importe quelles fenêtres de l'Éditeur de scripts qui pourraient être ouvertes) et l'Éditeur de scripts est toujours l'application active.

```
tell application "AppleWorks"
   set leCompteur to number of words of text body of ¬
      front document
end tell
if leCompteur > 500 then
   display dialog "Vous avez dépassé la limite autorisée."
end
```

Vous pouvez envoyer des commandes de compléments de pilotage à une cible située sur n'importe quel ordinateur distant, pourvu que le dossier "Compléments de pilotage" de l'ordinateur distant comporte le complément de pilotage approprié. Ceci est vrai également, même si sur l'ordinateur qui exécute le script, le complément de pilotage approprié n'est pas disponible dans le dossier "Compléments de pilotage". Par exemple, vous pouvez envoyer la commande Display Dialog, à n'importe quelle application située sur un ordinateur distant, possédant le complément de pilotage approprié, même si sur votre ordinateur ce complément de pilotage n'est pas disponible.

Chaque complément de pilotage contenant des gestionnaires de commandes a son propre dictionnaire, lequel liste les mots réservés - comprenant les noms des commandes, les étiquettes des paramètres et dans certains cas, les noms des objets - utilisés avec ses commandes. Si un dictionnaire d'un complément de pilotage comporte des termes qui font aussi partie du dictionnaire d'une application, vous ne pourrez pas utiliser ces termes à l'intérieur de l'instruction Tell de cette application.

Si vous spécifiez un script-objet comme la cible d'une commande d'un complément de pilotage, le script-objet, soit gère lui-même la commande (parfois en la modifiant), soit il utilise une instruction Continue pour transmettre la commande à l'application cible par défaut. Pour plus d'informations sur les compléments de pilotage, les scripts-objets et l'instruction Continue, voir le tome 7 "Les scrips-objets" du *Guide AppleScript version française*.

Chapitre 4

Les définitions des commandes

Ce chapitre décrit le fonctionnement et l'utilisation des commandes du complément de pilotage "Compléments standard" fourni avec AppleScript.

Les définitions des commandes sont classées par ordre alphabétique sur leur nom. Pour plus d'informations sur l'utilisation des définitions de commandes, voir le tome 2 du *Guide AppleScript version française.*

➥ **Note des traducteurs francophones**
Notez que les messages d'erreur, listés dans "Erreurs", peuvent varier avec les versions d'AppleScript, d'AppleEvent et de Mac OS. Les messages listés dans les tableaux, correspondent à Mac OS 8.6 et AS 1.3.7. ●

Activate

La commande Activate amène une application à l'avant-plan (ses fenêtres passent au premier plan sur le bureau). Si l'application est locale, AppleScript ouvre l'application si elle n'est pas déjà lancée. Si elle est sur un ordinateur distant, elle doit déjà être lancée, la commande Activate n'ouvrira pas l'application. La commande Activate fait partie, soit du dialecte anglais, soit de l'extension AppleScript, suivant la version de Mac OS et d'AS. Le dictionnaire d'AppleScript n'est pas accessible depuis l'Éditeur de scripts, mais par contre depuis Smile 1.8.4 Fr.

Syntaxe

`activate` *referenceToApplication*

Paramètres

referenceToApplication	Une référence de la forme `application` *nameString* (voir "Notes"). *Classe :* Reference

Résultats

Aucun

Exemples

```
set x to application "AppleWorks"
activate x

activate application "Disque Dur:Applications:AppleWorks"

tell application "AppleWorks"
   activate
end tell

tell application "AppleWorks" to activate
```

Notes

La manière de spécifier le nom (*nameString*) de l'application à activer, dépend si l'application est locale ou sur un ordinateur distant.

Pour spécifier une application locale, utilisez une chaîne de caractères telle que "*Disque*:*Dossier1*:*Dossier2*:. . . . :*nomApplication*". Vous pouvez aussi spécifier une chaîne de caractères avec seulement le nom de l'application ("*nomApplication*"). Dans ce cas, AppleScript essaie de trouver l'application dans le répertoire courant.

Pour spécifier une application sur un ordinateur distant, vous devez utiliser une chaîne de caractères indiquant le nom de l'application tel qu'il est écrit dans le menu Application ("*nomApplication*"), et vous devez aussi spécifier le nom de l'ordinateur et si nécessaire la zone dans laquelle l'ordinateur est situé. L'application doit déjà être lancée. La commande Activate ne lance pas les applications situées sur des ordinateurs distants.

Pour plus d'informations sur les références aux applications, voir le tome 3 du *Guide AppleScript version française.*

Erreurs

Numéro	Message d'erreur
-600	L'application n'est pas ouverte.
-606	L'application fonctionne en tâche de fond uniquement.

ASCII Character

La commande ASCII Character retourne le caractère ASCII associé au nombre spécifié.

Syntaxe

`ASCII Character` *integer*

Paramètres

integer	Une expression qui évalue un entier compris entre 1 et 255. *Classe :* Integer

Résultats

Le caractère qui correspond au nombre ASCII spécifié.

Exemples

```
ASCII character 100
-- résultat : "d"

ASCII character 101
-- résultat : "e"
```

Erreurs

Numéro	Message d'erreur
-108	Mémoire saturée.
-1700	Impossible de faire de <data> un item.

-1701	Le paramètre <name> est manquant pour <commandName>.
-1704	Certains paramètres sont invalides.
-1705	L'opération impliquant un élément liste a échoué.
-1718	La réponse n'est pas encore arrivée.
-1720	Plage invalide.

ASCII Number

La commande ASCII Number retourne le nombre ASCII associé au caractère spécifié.

Syntaxe

`ASCII number` *string*

Paramètres

string	Un caractère ASCII. *Classe :* String

Résultats

Le nombre ASCII qui correspond au caractère spécifié.

Exemples

```
ASCII number "d"
-- résultat : 100
ASCII number "e"
-- résultat : 101
```

Erreurs

Numéro	Message d'erreur
-108	Mémoire saturée.
-1700	Impossible de faire de <data> un item.

-1701	Le paramètre <name> est manquant pour <commandName>.
-1704	Certains paramètres sont invalides.
-1705	L'opération impliquant un élément liste a échoué.
-1715	Certains paramètres n'ont pas été compris.
-1718	La réponse n'est pas encore arrivée.
-1720	Plage invalide.

Beep

La commande Beep joue le son d'alerte du Macintosh.

Syntaxe

```
beep [ numberOfBeeps ]
```

Paramètres

numberOfBeeps Le nombre de fois que le son d'alerte doit être joué. Si vous omettez *numberOfBeeps*, le son d'alerte est joué une seule fois.
Classe : Integer
Valeur par défaut : 1

Résultats

Aucun

Exemples

```
beep
beep 3
```

Notes

L'utilisateur peut annuler une commande Beep - par exemple, si la valeur du paramètre *numberOfBeeps* est trop important - en appuyant simultanément sur Cmd + . (point) ou sur Esc.

Choose Application

La commande Choose Application permet à l'utilisateur de choisir une application, parmi celles déjà lancées, depuis une boîte de dialogue.

La boîte de dialogue affiche les applications de l'ordinateur local et celles de n'importe quel ordinateur connecté au même réseau.

Syntaxe

```
choose application [ with prompt promptString ] ¬
   [ application label appListLabel ]
```

Paramètres

promptString Le texte qui doit être affiché dans l'en-tête de la boîte de dialogue. Le texte peut comporter jusqu'à 255 caractères, mais la boîte de dialogue ne peut en afficher que 50 dans le champ prévu. Si vous omettez le paramètre `with prompt`, la chaîne de caractères "Établir un lien avec :" sera affichée par défaut.
Classe : String
Valeur par défaut : "Établir un lien avec :"

appListLabel Le texte qui doit être affiché au-dessus de la fenêtre listant les applications disponibles. Le texte peut comporter jusqu'à 255 caractères, mais la boîte de dialogue ne peut en afficher que 25 dans le champ prévu. Si vous omettez le paramètre `application label`, la chaîne de caractères `"Applications"` sera affichée par défaut.
Classe : String
Valeur par défaut : "Applications"

Résultats

Une référence à l'application spécifiée par l'utilisateur.

Exemples

```
choose application with prompt "Choisissez une application :"
```

```
tell (choose application with prompt ¬
   "Choisissez un traitement de texte :")
   -- diverses instructions
end tell
```

Notes

Si l'utilisateur choisit le bouton Annuler, Choose Application retourne une erreur numéro -128 (Annulé par l'utilisateur). Si vous voulez que votre script continue l'exécution après que l'utilisateur ait cliqué sur le bouton Annuler, vous devez fournir un gestionnaire d'erreur pour l'erreur -128. Pour plus d'informations sur les instructions Tell et les gestionnaires d'erreur, voir le tome 5 "Les instructions de contrôle" du *Guide AppleScript version française.*

Erreurs

Numéro	Message d'erreur
-108	Mémoire saturée.
-128	Annulé par l'utilisateur.

Choose File

La commande Choose File affiche une boîte de dialogue dans laquelle l'utilisateur choisit un fichier.

Syntaxe

`choose file [ with prompt` *promptString* `] [ of type` *listOfTypes* `]`

Paramètres

promptString Le texte qui doit être affiché dans la boîte de dialogue. Le texte peut comporter jusqu'à 255 caractères, mais la boîte de dialogue ne peut en afficher que 40 dans le champ prévu. Si vous omettez le paramètre `with prompt`, aucun texte ne sera affiché.
Classe : String
Valeur par défaut : (aucun texte)

listOfTypes Une liste des types de fichiers qui doivent être affichés dans la boîte de dialogue. Chaque type est indiqué par une chaîne de 4 caractères correspondant au code du type de fichiers voulus, comme `"TEXT"`, `"APPL"`, `"PICT"` ou `"PNTG"`. Si vous omettez le paramètre `of type`, tous les fichiers seront affichés.
Classe : List de strings ; chaque chaîne est codée sur 4 caractères
Valeur par défaut : (tous les types de fichiers sont affichés)

Résultats

Une référence telle que `file "`*Disque Dur*`:`*Dossier1*`:`*Dossier2*`...:`*nomFichier*`"` pour le fichier spécifié par l'utilisateur, si un fichier a été choisi.

➥ **Note des traducteurs francophones**
Une référence telle que `alias "`*Disque Dur*`:`*Dossier1*`:`*Dossier2*`...:`*nomFichier*`"` pour les versions supèrieures à OS 8.6 ; AS 1.3.7. ●

Exemples

```
choose file with prompt "Choisissez un fichier :" of type ¬
   {"TEXT","APPL"}
open result
```

Notes

Si l'utilisateur clique sur le bouton Annuler dans la boîte de dialogue de Choose File, AppleScript retourne une erreur numéro -128 (Annulé par l'utilisateur). Si vous voulez que votre script continue l'exécution après que l'utilisateur ait cliqué sur le bouton Annuler, vous devez fournir un gestionnaire d'erreur pour l'erreur -128. Pour plus d'informations sur les instructions Try et les gestionnaires d'erreur, voir le tome 5 "Les instructions de contrôle" du *Guide AppleScript version française.*

Erreurs

Numéro	Message d'erreur
-108	Mémoire saturée.
-128	Annulé par l'utilisateur.

Choose Folder

La commande Choose Folder affiche une boîte de dialogue dans laquelle l'utilisateur peut choisir un répertoire (un dossier, un volume ou le bureau).

Syntaxe

```
choose folder [ with prompt promptString ]
```

Paramètres

promptString Le texte qui apparaît dans la boîte de dialogue. Le texte peut comporter jusqu'à 255 caractères, mais la boîte de dialogue ne peut en afficher que 80 dans le champ prévu. Si vous omettez le paramètre `with prompt`, la chaîne `"Choisissez un dossier :"` sera affichée par défaut.
Classe : String
Valeur par défaut : "Choisissez un dossier :"

Résultats

Une référence telle que `file "`*Disque Dur*`:`*Dossier1*`:`*Dossier2*`...:`*nomDossier*`:"` pour le répertoire spécifié par l'utilisateur, si un répertoire a été choisi.

➥ **Note des traducteurs francophones**
Une référence telle que `alias "`*Disque Dur*`:`*Dossier1*`:`*Dossier2*`...:`*nomFichier*`"` pour les versions supèrieures à OS 8.6 ; AS 1.3.7. ●

Exemples

```
choose folder with prompt "Choisissez un volume :"
```

Notes

Si l'utilisateur clique sur le bouton Annuler dans la boîte de dialogue de Choose Folder, AppleScript retourne une erreur numéro -128 (Annulé par l'utilisateur). Si vous voulez que votre script continue l'exécution après que l'utilisateur ait cliqué sur le bouton Annuler, vous devez fournir un gestionnaire d'erreur pour l'erreur -128. Pour plus d'informations sur les

instructions Try et les gestionnaires d'erreur, voir le tome 5 "Les instructions de contrôle" du *Guide AppleScript version française.*

Erreurs

Numéro	Message d'erreur
-108	Mémoire saturée.
-128	Annulé par l'utilisateur.

Choose from List

Définition rajoutée par rapport au guide Apple

La commande Choose from List permet de présenter à l'utilisateur une liste d'éléments, parmi lesquels, il peut choisir un ou plusieurs éléments, ou aucun.

Syntaxe

```
choose from list listOfElementsToChoose ¬
   [ with prompt promptString ] ¬
   [ default items defaultListOfElements ] ¬
   [ OK button nameOkButton ] ¬
   [ cancel button name nameCancelButton ] ¬
   [ multiple selections allowed booleanMultipleSelection] ¬
   [ empty selection allowed booleanEmptySelection]
```

Paramètres

listOfElementsToChoose — Une liste d'éléments à choisir. Les éléments de la liste peuvent appartenir indifféremment aux classes String, Integer, Real, List à élément unique ou des expressions booléennes. La liste d'éléments ne doit pas être vide.
Classe : List

promptString Le texte qui apparaît dans la boîte de dialogue. Le texte peut comporter jusqu'à 255 caractères. La fenêtre affichant les choix se proportionnera en conséquence. Si vous omettez le paramètre `with prompt`, la chaîne "Veuillez faire votre choix :" sera affichée par défaut.
Classe : String
Valeur par défaut : "Veuillez faire votre choix :"

defaultListOfElements Les éléments de la liste qui sont sélectionnés par défaut. Si vous omettez le paramètre `default items`, aucun élément ne sera pré-sélectionné.
Classe : Liste

nameOkButton Le texte qui apparaît dans le bouton de validation. Pour la longueur maximale à respecter, voir "Notes". Si vous omettez le paramètre `OK button name`, "Ok" apparaîtra par défaut dans le bouton de validation.
Classe : String
Valeur par défaut : "Ok"

nameCancelButton Le texte qui apparaît dans le bouton d'annulation. Pour la longueur maximale à respecter, voir "Notes". Si vous omettez le paramètre `cancel button name`, "Annuler" apparaîtra par défaut.
Classe : String
Valeur par défaut : "Annuler"

booleanMultipleSelection Une expression qui vaut `true` ou `false`. Si elle vaut `true`, vous pourrez choisir plusieurs éléments. La sélection multiple se fait en maintenant appuyée, soit la touche maj, soit la touche Cmd, lorsque que vous cliquez sur les éléments choisis. Si elle vaut `false` ou si vous omettez ce paramètre, vous ne pourrez choisir qu'un seul élément.
Classe : Boolean
Valeur par défaut : false

booleanEmptySelection Une expression qui vaut `true` ou `false`. Si elle vaut `true`, vous ne serez pas obligé de choisir un élément de la liste pour valider la boîte de dialogue. Si elle vaut `false` ou si vous omettez ce paramètre, vous devrez obligatoirement choisir un élément pour pouvoir valider la boîte de dialogue.
Classe : Boolean
Valeur par défaut : false

Résultats

La liste des éléments sélectionnés. Notez que la classe de valeur d'origine des éléments est conservée. Par exemple, si vous choisissez un élément de la classe Real, le résultat retourné, sera une liste contenant l'élément de la classe Real.

Si le paramètre `empty selection allowed` est réglé sur `true`, vous pouvez valider sans avoir choisi un élément de la liste, le résultat sera alors une liste vide.

Si vous choisissez d'annuler la boîte de dialogue, un résultat sera retourné, ce sera la valeur booléenne `false`.

Exemples

```
choose from list {1, "2", {"a"}, 2.36, "Toto", 5>4} ¬
   with prompt "Cliquez dans la liste :" ¬
   default items {2.36, 5>4} ok button name "Validation" ¬
   cancel button name "Arrêt" ¬
   multiple selections allowed true ¬
   empty selection allowed 18>1
```

Notes

Afin d'éviter les problèmes d'affichage pour les boutons, il ne faut pas que la taille totale des textes des deux boutons réunis dépassent 28 caractères. C'est à dire 14 caractères pour l'un, 14 pour l'autre ou, 5 pour l'un, 23 pour l'autre.

Notez que les phrases `with multiple selections allowed` et `with empty selection allowed` sont équivalentes à respectivement, `multiple selections allowed true` et `empty selection allowed true`. De même,

`without multiple selections allowed` et `without empty selection allowed` sont équivalentes à, respectivement, `multiple selections allowed false` et `empty selection allowed false`.

Erreurs

Numéro	Message d'erreur
-108	Mémoire saturée.

Clipboard Info

Définition rajoutée par rapport au guide Apple

La commande Clipboard Info fournit une description du contenu du presse-papiers, non le contenu en tant que tel.

Syntaxe

```
clipboard info [ for typeClass ]
```

Paramètres

typeClass Le type de données souhaitées. Si ce paramètre est omis, la commande Clipboard Info retournera toutes les informations disponibles.
Classe : Class

Résultats

Une liste de listes telle que {{type de données, taille des données}}

Exemples

```
tell application "Finder"
   clipboard info
end tell
-- résultat : {{string,29},{styled clipboard text,62}}
```

Notes

La commande Clipboard Info fournit une description du contenu du presse-papiers de l'application active. Si plus d'une application est ouverte, utilisez la commande Activate pour mettre l'application désirée au premier-plan.

La commande Clipboard Info peut gérer différents types de données. Le texte stylé est le type de données par défaut, le texte standard est utilisé comme texte de sauvegarde. Les caractères internationaux, les images et les sons sont également gérés.

Close Access

La commande Close Access ferme un fichier qui avait été ouvert avec la commande Open for Access. Pour plus d'informations sur ces commandes, voir "Utiliser les commandes Read / Write" (page 76).

Syntaxe

`close access` *referenceToFile*

Paramètres

referenceToFile Une référence de la forme `file` *nameString* ou `alias` *nameString*, ou un numéro de référence de fichier obtenu précédemment avec la commande Open for Access (voir "Notes").
Classe : Reference ou Integer

Résultats

Aucun

Exemples

Cet exemple fermera le fichier nommé Toto à l'emplacement spécifié, si celui-ci a été précédemment ouvert avec la commande Open for Access.

```
tell application "AppleWorks"
   close access file "Disque Dur:Blagues:Toto"
```

```
end tell
```

L'exemple suivant fermera le fichier associé au numéro de référence de fichier, si la valeur de `myFileRefNumber` est un numéro de référence de fichier précédemment obtenu avec la commande Open for Access.

```
tell application "AppleWorks"
   close access myFileRefNumber
end tell
```

Notes

Pour spécifier le nom (*nameString*) d'un fichier, utilisez une chaîne de caractères de la forme "*Disque Dur*:*Dossier1*:*Dossier2*. . . :*nomFichier*" comme il est décrit dans le tome 3 "Les objets et les références" du *Guide AppleScript version française*. Si vous indiquez uniquement le nom du fichier (*nomFichier*) au lieu du chemin en entier, AppleScript essaiera de trouver le fichier dans le répertoire courant.

Si vous spécifiez une référence à un fichier ou à un alias, la commande Close Access essaiera de faire correspondre la référence avec un fichier précédemment ouvert avec la commande Open for Access. Si une correspondance est trouvée, la commande Close Access fermera le fichier. Si aucune correspondance n'est trouvée, la commande Close Access retournera une erreur numéro -43 (Le fichier <name> est introuvable).

Si vous spécifiez un numéro de référence de fichier précédemment obtenu avec la commande Open for Access, la commande Close Access fermera immédiatement le fichier spécifié.

Erreurs

Numéro	Message d'erreur
-35	Le disque <name> est introuvable.
-38	Le fichier <name> n'a pas été ouvert.
-43	Le fichier <name> est introuvable.
-50	Erreur de paramètre.

Current Date

La commande Current Date retourne une valeur date correspondant à l'heure, le jour et la date courante.

Syntaxe

```
current date
```

Résultats

Une valeur date.

Exemples

```
set timeOfTransfert to current date
get timeOfTransfert

if date string of (current date) = ¬
   date string of (date "01/05/2002") then
   display dialog "Aujourd'hui, férié"
else
   display dialog "Aujourd'hui, pas férié"
end if
```

Erreurs

Numéro	Message d'erreur
-108	Mémoire saturée.
-1700	Impossible de faire de <data> un item.
-1701	Le paramètre <name> est manquant pour <commandName>.
-1704	Certains paramètres sont invalides.
-1705	L'opération impliquant un élément liste a échoué.
-1718	La réponse n'est pas encore arrivée.

Delay

Définition rajoutée par rapport au guide Apple

La commande Delay permet de marquer une pause, en secondes, pendant l'exécution d'un script.

Syntaxe

`delay [` *numberOfSeconds]*

Paramètres

numberOfSeconds Le délai en secondes avant la reprise de l'exécution.
Classe : Integer

Résultats

Aucun

Exemples

```
delay 4 -- pause de 4 secondes
```

Notes

L'activité des applications en arrière-plan n'est pas affectée par la commande Delay.

Display Dialog

La commande Display Dialog affiche une boîte de dialogue. Cette boîte de dialogue contient une chaîne de caractères et un ou plusieurs boutons, comme Oui, Non, Ok et Annuler. La boîte de dialogue peut aussi contenir un icone et un champ de saisie dans lequel l'utilisateur peut saisir du texte.

Syntaxe

```
display dialog questionString                    ¬
   [ default answer answerString ]               ¬
   [ buttons buttonList ]                        ¬
   [ default button buttonNumberOrName ]         ¬
   [ with icon iconNumberOrName ]                ¬
   [ giving up after numberOfSeconds ]
```

Paramètres

questionString — Le texte affiché dans la boîte de dialogue. Le texte peut comporter jusqu'à 255 caractères.
Classe : String

answerString — Le texte par défaut du champ de saisie dans lequel l'utilisateur peut saisir du texte. Le texte peut comporter jusqu'à 255 caractères. Si vous omettez le paramètre `default answer`, la boîte de dialogue ne comportera pas de champ de saisie. Si vous spécifiez une chaîne vide (""), la boîte de dialogue aura un champ de saisie sans texte par défaut, vide.
Classe : String

buttonList — La liste des boutons qui doivent apparaître dans la boîte de dialogue. Chaque élément de la liste est une chaîne contenant le texte qui apparaît dans le bouton. La première chaîne de la liste est le texte pour le bouton affiché le plus à gauche dans la boîte de dialogue, la seconde chaîne est pour le bouton suivant, et ainsi de suite. Vous pouvez spécifier jusqu'à trois boutons maximum. Si vous omettez le paramètre `buttons`, le dialogue contiendra deux boutons : Annuler et Ok.
Classe : List of Strings

buttonNumberOrName Le bouton activé par défaut. Vous pouvez spécifier le bouton activé par défaut avec une chaîne de caractères (la chaîne fournie pour le bouton dans *buttonList*), ou avec un nombre entier (lequel spécifie la position du bouton dans *buttonList* ; 1 indique le premier bouton en partant de la gauche, 2 indique le deuxième bouton en partant de la gauche, et ainsi de suite).
Classe : String ou Integer

iconNumberOrName L'icone qui doit être affiché dans la boîte de dialogue. Cela peut être soit une chaîne de caractères spécifiant le nom d'une ressource 'ICON', ou un nombre entier spécifiant le numéro ID de la ressource (voir "Notes").
Classe : String

numberOfSeconds Temps en secondes imparti avant que la boîte de dialogue ne se referme automatiquement.
Classe : Integer

Résultats

Un enregistrement de la classe d'objet Dialog Reply (défini par le complément de pilotage "Compléments standard") qui contient les propriétés suivantes :

`button returned` Le texte du bouton que l'utilisateur a pressé pour fermer la boîte de dialogue.
Classe : String

`text returned` Le texte du champ de saisie de la boîte de dialogue. Si le dialogue ne comporte pas de champ de saisie, rien n'est retourné. S'il n'y a pas de texte saisi dans le champ, la valeur retournée est une chaîne vide ("").
Classe : String

`gave up` Cette propriété n'est retournée que si le paramètre `Giving up After` était explicitement spécifié dans la définition de Display Dialog. La valeur retournée est `true` si l'utilisateur a répondu avant l'expiration du délai. La valeur retournée est `false` si l'utilisateur n'a pas répondu.
Classe : Boolean

Exemples

L'exemple suivant utilise la commande Display Dialog, il permet à l'utilisateur de saisir un mot de passe.

```
set prompt to "Entrez votre mot de passe :"
repeat with i from 1 to 4
   set dialogResult to display dialog prompt ¬
       buttons {"J'abandonne", "Ok"} default button 2 ¬
       default answer "toto" with icon 1
   if button returned of dialogResult = "J'abandonne" then
       error number -128 --Annulé par l'utilisateur
   else
       if text returned of dialogResult is "tutu" then
           exit
       else
           if i < 3 then
             set prompt to "Pas bon !. Recommencez" & i + 1
           else
             set prompt to "Dernière chance !. Recommencez"
           end if
       end if
   end if
end repeat
```

Notes

La taille de la boîte de dialogue est déterminée par la longueur des chaînes de caractères de la question, de la réponse et des boutons.

Si vous trouvez que le champ de saisie pour la réponse est trop court, vous pouvez l'agrandir comme dans cet exemple :

```
display dialog "Tapez votre demande " ¬
      default answer "\r\r\r\r\r"
-- la réponse peut alors être sasie sur 5 lignes sans masquage
```

Comme alternative au bouton Annuler, l'utilisateur peut appuyer sur les touches Cmd + . (point) ou sur la touche Esc, pour annuler la boîte de dialogue affichée par la commande Display Dialog.

Une boîte de dialogue peut afficher un icone qui vient, soit du fichier du script, soit de l'application courante (comme celle qui est spécifiée par une instruction Tell), soit du fichier Système. S'il y a un icone avec le nom ou le numéro spécifié dans le fichier du script, il sera utilisé ; sinon, AppleScript vérifie l'application courante ; finalement, si l'icone n'est trouvé ni dans le

fichier du script, ni dans l'application courante, AppleScript vérifie le fichier Système. Pour ajouter des icones à un fichier d'un script ou d'une application, utilisez un éditeur de ressources comme ResEdit.

Le fichier Système fournit trois icones standards pour avertir l'utilisateur d'un problème. Ces icones, voir les dessins plus bas, ont les significations suivantes :

- Stop (icone numéro 0). Utilisez cette icone pour attirer l'attention de l'utilisateur sur un grave problème.

- Avertissement (icone numéro 1). Utilisez cette icone pour fournir des informations.

- Attention (icone numéro 2). Utilisez cette icone pour avertir l'utilisateur que l'opération demandée peut avoir de graves conséquences irréversibles.

Les trois icones fournis par le fichier Système :

0

1

2

Pour utiliser ces icones, référez-vous y avec le numéro. Par exemple, ce script affiche l'icone Attention.

```
display dialog "" with icon 2
```

➥ **Note des traducteurs francophones**
iconNumberOrName peut aussi être stop, note ou caution. Respectivement équivalent à 0, 1, 2 (voir plus haut). À éviter néanmoins pour des questions de compatibilité entre différentes versions d'AS. L'utilisation des numéros étant préférable. ●

Erreurs

Numéro	Message d'erreur
-108	Mémoire saturée.
-128	Annulé par l'utilisateur.

-192	Une ressource est introuvable.
-1712	Délai dépassé pour un AppleEvent.

Get EOF

La commande Get EOF retourne un nombre entier indiquant, en octets, l'emplacement de la fin du fichier spécifié par rapport au début de ce fichier. Pour plus d'informations, voir "Utiliser les commandes Read/Write" (page 76).

Syntaxe

`get eof` *referenceToFile*

Paramètres

referenceToFile Une référence de la forme `file` *nameString* ou `alias` *nameString*, ou un numéro de référence de fichier retourné précédemment par un appel de la commande Open for Access (voir "Notes").
Classe : Reference ou Integer

Résultats

Si la commande réussit, elle retourne un nombre entier indiquant l'emplacement de la fin du fichier spécifié.

Exemples

```
get eof file "Disque Dur:Essai:Essai 2"
```

Notes

Pour spécifier le nom (*nameString*) d'un fichier, utilisez une chaîne de caractères de la forme "*Disque Dur*:*Dossier1*:*Dossier2*. . . :*nomFichier*" comme il est décrit dans le tome 3 "Les objets et les références" du *Guide AppleScript version française*. Si vous indiquez uniquement le nom du fichier (*nomFichier*) au lieu du chemin en entier, AppleScript essaiera de trouver le fichier dans le répertoire courant.

Si vous spécifiez une référence à un fichier ou à un alias, la commande Get EOF essaiera de faire correspondre la référence avec un fichier précédemment ouvert avec la commande Open for Access. Pour que cette commande fonctionne, l'ouverture de l'accès du fichier avec la commande Open for Access n'est pas obligatoire. Si une correspondance est trouvée, la commande Get EOF retournera un nombre entier, indiquant l'emplacement de la fin du fichier. Si aucune correspondance n'est trouvée, Get EOF ouvrira le fichier, obtiendra la position de la fin du fichier, puis le refermera.

Si vous spécifiez un numéro de référence de fichier précédemment obtenu avec la commande Open for Access, la commande Get EOF retournera le résultat immédiatement.

Erreurs

Numéro	Message d'erreur
-38	Le fichier <name> n'a pas été ouvert.
-50	Erreur de paramètre.
-51	Erreur de numéro de référence de fichier.

Info For

La commande Info For retourne un enregistrement contenant des informations sur un fichier ou un dossier spécifié.

Syntaxe

`info for` *referenceToFile*

Paramètres

referenceToFile Une référence de la forme `file` *nameString* ou `alias` *nameString*.
Classe : Reference

Résultats

Un enregistrement de la classe d'objet File Information contient les propriétés

suivantes :

`name`	Nom du fichier ou du dossier. *Classe* : String
`creation date`	Date et heure de création du fichier ou du dossier. *Classe* : Date
`modification date`	Date et heure de la dernière modification du fichier ou du dossier. *Classe* : Date
`icon position`	Emplacement du coin supérieur gauche de l'icone du fichier ou du dossier. *Classe* : Point (défini par une liste de deux entiers représentant les coordonnées verticales et horizontales)
`visible`	Une valeur qui indique si l'icone du fichier ou du dossier est visible sur le bureau (`true`) ou non (`false`). *Classe* : Boolean *Modifiable ?* Non
`size`	La taille du fichier ou du dossier en octets. *Classe* : Integer *Modifiable ?* Non
`folder`	Une valeur qui indique si l'objet décrit par l'enregistrement de la commande Info For est un dossier (`true`) ou un fichier (`false`). *Classe* : Boolean *Modifiable ?* Non
`alias`	Indique si le fichier est un alias (`true`) ou non (`false`). *Classe* : Boolean *Modifiable ?* Non
`folder window`	4 nombres entiers qui indiquent les coordonnées de la fenêtre du dossier. Retourné uniquement par les dossiers. *Classe* : Bounding rectangle (une liste de 4 nombres entiers) *Modifiable ?* Non
`locked`	Indique si le fichier ou le dossier est verrouillé (`true`) ou non (`false`). *Classe* : Boolean *Modifiable ?* Non

`file creator` Code créateur de 4 caractères du fichier. Retourné uniquement par les fichiers.
Classe : String
Modifiable ? Non

`file type` Code de 4 caractères identifiant le type de fichier du fichier. Retourné uniquement par les fichiers.
Classe : String
Modifiable ? Non

`short version` La version courte du fichier, s'il y a. Retourné uniquement par les fichiers.
Classe : String
Modifiable ? Non

`long version` La version longue du fichier, s'il y a. Retourné uniquement par les fichiers.
Classe : String
Modifiable ? Non

`default application` L'application par défaut pour un fichier qui n'est pas une application. Utilisé uniquement par les fichiers qui ne sont pas des applications.
Classe : Alias
Modifiable ? Non

`busy status` Propriété apparue avec Mac OS 9.0 ; AS 1.4.0. Renvoie la valeur `true` si le fichier spécifié par la commande est en cours d'utilisation.
Classe : Boolean
Modifiable ? Non

Exemples

Cet exemple retourne l'enregistrement entier de la commande Info For de l'Éditeur de scripts :

```
info for file ¬
"Disque Dur:Compléments Apple:AppleScript:Éditeur de scripts"
(* résultat : {name:"Éditeur de scripts", creation date:date
"vendredi 30 mai 1997 12:00:00", modification date:date
"vendredi 26 avril 2002 9:36:44", icon position:{188, 12},
visible:true, size:246243, folder:false, alias:false,
locked:false, file creator:"ToyS", file type:"APPL", short
version:"FU2-1.1.3", long version:"FU2-1.1.3, Copyright Apple
Computer, Inc. 1993-97"} *)
```

Cet exemple retourne juste le type de fichier de l'Éditeur de scripts :

```
set x to info for file ¬
"Disque Dur:Compléments Apple:AppleScript:Éditeur de scripts"
x's file type
-- résultat : "APPL"
```

Cet exemple affiche une boîte de dialogue si l'Éditeur de scripts n'est pas verrouilé :

```
if locked of (info for file ¬
"Disque Dur:Compléments Apple:AppleScript:Éditeur de scripts")¬
is false then
   display dialog "L'Éditeur de scripts n'est pas verrouillé"
end if
```

Notes

Pour spécifier le nom (*nameString*) d'un fichier, utilisez une chaîne de caractères de la forme "*Disque Dur*:*Dossier1*:*Dossier2*. . . :*nomFichier*" comme il est décrit dans le tome 3 "Les objets et les références" du *Guide AppleScript version française*. Si vous indiquez uniquement le nom du fichier (*nomFichier*) au lieu du chemin en entier, AppleScript essaiera de trouver le fichier dans le répertoire courant.

Erreurs

Numéro	Message d'erreur
-35	Le disque <name> est introuvable.
-37	Mauvais nom de fichier.
-43	Le fichier <name> est introuvable.
-50	Erreur de paramètre.
-108	Mémoire saturée.
-120	Le dossier <name> est introuvable.
-1700	Impossible de faire de <data> un item.
-1701	Le paramètre <name> est manquant pour <commandName>.

-1704	Certains paramètres sont invalides.
-1705	L’opération impliquant un élément liste a échoué.
-1718	La réponse n’est pas encore arrivée.

List Disks

La commande List Disks retourne une liste contenant les noms de chaque volume monté, y compris les CD-Roms, les disquettes, les volumes AppleShare et plus.

Syntaxe

```
list disks
```

Résultats

Une liste de chaînes de caractères.

Exemples

```
list disks
-- résultat : {"Disque Dur", "Sauvegarde", "CD Audio 1"}
```

Erreurs

Numéro	Message d’erreur
-35	Le disque <name> est introuvable.
-50	Erreur de paramètre.
-108	Mémoire saturée.
-120	Le dossier <name> est introuvable.
-1703	<data> est le mauvais type.
-1704	Certains paramètres sont invalides.
-1705	L’opération impliquant un élément liste a échoué.
-1719	Impossible d’obtenir <reference>. Index invalide.

List Folder

La commande List Folder retourne une liste de tous les fichiers et de tous les dossiers contenus dans un dossier ou sur un volume.

Syntaxe

`list folder` *referenceToFolder* `[ invisibles` *boolean* `]`

Paramètres

referenceToFolder Une référence de la forme `file` *nameString*, `alias` *nameString*, ou `folder` *nameString* (voir “Notes”).
Classe : Reference ou String

boolean Apparu avec Mac OS 8.6 ; AS 1.3.7. Une expression qui vaut `true` ou `false`. Si elle vaut `true`, les fichiers invisibles sont listés.
Classe : Boolean
Valeur par défaut : true

Résultats

Une liste de chaînes de caractères.

Exemples

```
list folder "Sauvegarde:"
-- résultat : {"Lettres", "Traduction AppleScript"}
```

Notes

Pour spécifier le nom (*nameString*) d’un dossier, utilisez une chaîne de caractères de la forme "*Disque Dur*:*Dossier1*:*Dossier2*. . . :*nomDossier*:" comme il est décrit dans le tome 3 “Les objets et les références” du *Guide AppleScript version française.* Si vous indiquez uniquement le nom du dossier (*nomDossier*) au lieu du chemin en entier, AppleScript essaiera de trouver le dossier dans le répertoire courant.

Erreurs

Numéro	Message d'erreur
-35	Le disque <name> est introuvable.
-37	Mauvais nom de fichier.
-43	Le fichier <name> est introuvable.
-50	Erreur de paramètre.
-108	Mémoire saturée.
-120	Le dossier <name> est introuvable.
-1700	Impossible de faire de <data> un item.
-1701	Le paramètre <name> est manquant pour <commandName>.
-1703	<data> est le mauvais type.
-1704	Certains paramètres sont invalides.
-1705	L'opération impliquant un élément liste a échoué.
-1718	La réponse n'est pas encore arrivée.
-1719	Impossible d'obtenir <reference>. Index invalide.

Load Script

La commande Load Script charge un script compilé dans le script courant comme un script-objet. Un script-objet est un objet défini par l'utilisateur qui est traité par AppleScript comme une valeur. Les scripts-objets sont décrits dans le tome 7 "Les scripts-objets" du *Guide AppleScript version française.*

Syntaxe

`load script` *referenceToFile*

Paramètres

referenceToFile Une référence de la forme `file` *nameStrin* ou `alias` *nameString* (voir "Notes"). Le fichier doit être un script compilé ou un script-application. Il ne doit pas être un fichier texte.
Classe : Reference

Résultats

Un script-objet.

Exemples

L'exemple suivant charge un script compilé appelé Opérations Numériques et stocke le script-objet résultant dans la variable numericLib. L'instruction Tell montre comment appeler une routine contenue dans le script-objet.

```
load script file "Disque Dur:Scripts:Opérations Numériques"
set numericLib to result

tell numericLib
   factorial(10)
end tell
```

Notes

Pour spécifier le nom (*nameString*) d'un fichier, utilisez une chaîne de caractères de la forme "*Disque Dur*:*Dossier1*:*Dossier2*. . . :*nomFichier*" comme il est décrit dans le tome 3 "Les objets et les références" du *Guide AppleScript version française*. Si vous indiquez uniquement le nom du fichier (*nomFichier*) au lieu du chemin en entier, AppleScript essaiera de trouver le fichier dans le répertoire courant.

Pour plus d'informations sur l'utilisation de la commande Load Script pour sauvegarder et charger des librairies de routines, voir le tome 6 "Les gestionnaires" du *Guide AppleScript version française.*

Erreurs

Numéro	Message d'erreur
-108	Mémoire saturée.
-192	Une ressource est introuvable.
-1700	Impossible de faire de <data> un item.
-1701	Le paramètre <name> est manquant pour <commandName>.
-1703	<data> est le mauvais type.

-1704	Certains paramètres sont invalides.
-1705	L'opération impliquant un élément liste a échoué.
-1718	La réponse n'est pas encore arrivée.

Log

La commande Log reporte le résultat d'une ou de plusieurs variables entre les caractères (* et *) dans la fenêtre Session d'état de l'Éditeur de scripts. Log est une commande propriétaire pour l'Éditeur de scripts d'Apple. Elle fait partie, soit du dialecte anglais, soit de l'extension AppleScript, suivant la version de Mac OS et d'AS.

➥ **Note des traducteurs francophones**
À noter qu'à partir de la version de l'Éditeur de scripts livrée avec Mac OS 9.0, la fenêtre Session d'état a été renommée Historique des événements, mais son mode de fonctionnement est resté identique. ●

Syntaxe

```
log stringToLog
```

Paramètres

stringToLog	Une expression qui évalue une chaîne de caractères ou une valeur qui peut être contrainte en string. La chaîne de caractères résultante est affichée dans la fenêtre Session d'état de l'Éditeur de scripts. *Classe :* String

Résultats

Aucun

Exemples

```
log "Ce texte apparaîtra dans la fenêtre Session d'état"
```

Après l'exécution de cette instruction, le texte apparaîtra comme ceci :

```
(* Ce texte apparaîtra dans la fenêtre Session d'état *)
```

Notes

La commande Log fonctionne même si l'affichage des événements a été désactivé par la commande Stop Log. La commande Stop Log n'a aucune influence sur la commande Log.

Erreurs

Numéro	Message d'erreur
-1700	Impossible de faire de <data> un item.

Mount Volume

Définition rajoutée par rapport au guide Apple

La commande Mount Volume permet de monter des volumes dans un réseau AppleTalk ou AppleShare IP.

Syntaxe

```
mount volume nameVolume on server nameServer ¬
   [ in AppleTalk zone nameZone ] ¬
   [ as user name nameUser ] ¬
   [ with password passwordString ]
```

Paramètres

nameVolume	Pour une connexion AppleTalk, le nom du volume à monter. Pour une connexion AppleShare IP, l'URL du volume à monter. *Classe :* String
nameServer	Le nom de l'ordinateur serveur. *Classe :* String
nameZone	Le nom de la zone AppleTalk sur laquelle se situe le volume à monter. *Classe :* String

nameUser Le nom d'utilisateur pour une connexion AppleTalk. Si ce paramètre est omis et qu'un nom d'utilisateur est requis, une boîte de dialogue s'affichera.
Classe : String

passwordString Le mot de passe de la connexion AppleTalk. Si ce paramètre est omis et qu'un mot de passe est requis, une boîte de dialogue s'affichera.
Classe : String

Résultats

Aucun

Exemples

```
mount volume "Disque Dur" on server "Mon serveur" ¬
in Apple Talk zone "Blagues" as user name "Toto" ¬
with password "0+0"

mount volume ¬
"afp://mon.appleshare.volume.com/mon_dossier/"
```

New File (Choose File Name (OS 9.1 / AS 1.5.5))

La commande New File affiche une boîte de dialogue dans laquelle l'utilisateur peut spécifier un nom de fichier et un emplacement pour l'enregistrement. La commande New File ne crée pas un nouveau fichier ; elle retourne une référence de fichier avec le nom et l'emplacement spécifiés par l'utilisateur. Vous pouvez stocker la référence dans une variable et la transmettre à la commande Open for Access (laquelle en retour crée le fichier à l'emplacement indiqué), ou à n'importe quelle autre commande pour laquelle vous souhaitez spécifier un fichier qui n'existe pas encore.

Syntaxe

```
new file | choose file name (OS 9.1/AS 1.5.5)   ¬
   [ with prompt promptString ]                  ¬
   [ default name defaultName ]
```

Paramètres

promptString Le texte qui apparaît dans la boîte de dialogue. La chaîne de caractères spécifiée peut comporter jusqu'à 255 caractères. Si vous omettez le paramètre `with prompt`, le texte sera par défaut "Nouveau nom de fichier :".
Classe : String
Valeur par défaut : "Nouveau nom de fichier :"

defaultName Le nom du fichier par défaut qui apparaît dans la boîte de dialogue. La chaîne de caractères spécifiée peut comporter jusqu'à 255 caractères. Si vous omettez le paramètre default name, "Sans titre" sera affiché par défaut.
Classe : String
Valeur par défaut : "Sans titre"

Résultats

Une référence telle `file "`*Disque Dur*`:`*Dossier1*`:`*Dossier2*`:....:`*nomFichier*`"` pour le nom du fichier et l'emplacement spécifiés par l'utilisateur.

Exemples

```
set x to new file
open for access x
```

Notes

Si l'utilisateur choisit le bouton Annuler, la commande New File retourne une erreur -128. Si vous voulez que votre script continue l'exécution après que l'utilisateur ait cliqué sur Annuler, vous devrez spécifier explicitement un gestionnaire d'erreur. Pour plus d'informations sur les instructions Tell et les gestionnaires d'erreurs, voir le tome 5 "Les instructions de contrôle" du *Guide AppleScript version française.*

Erreurs

Numéro	Message d'erreur
-108	Mémoire saturée.
-128	Annulé par l'utilisateur.

Offset

La commande Offset retourne la position d'une chaîne de caractères dans une chaîne de caractères. Par exemple, la position de "Toto" dans "Les blagues à Toto" est 15, car la chaîne de caractères "Toto" commence avec le 15ème caractère du container. La position de "Les" dans "Les blagues à Toto" est 1, car "Les" commence avec le 1er caractère du container. La commande Offset est sensible à la casse.

Syntaxe

```
offset of stringToFind in stringToSearch
```

Paramètres

stringToFind	La chaîne de caractères à trouver dans *stringToSearch.* *Classe* : String
stringToSearch	Une chaîne de caractères dans laquelle la recherche s'effectue. *Classe :* String

Résultats

Le résultat est un nombre entier indiquant la position, en nombre de caractères, du premier caractère de *stringToFind* par rapport au début de *stringToSearch.* Si *stringToFind* n'est pas dans *stringToSearch,* AppleScript retourne la valeur 0.

Exemples

```
offset of "j" in "j'ai cinq doigts"
-- résultat : 1

offset of "doigts" in "j'ai cinq doigts"
résultat: 11

offset of "six" in "j'ai cinq doigts"
-- résultat : 0

offset of "Doigts" in "j'ai cinq doigts"
-- résultat : 0 -- à cause de la sensibilité à la casse
```

Notes

La commande Offset compare les chaînes de caractères, caractère par caractère, comme l'opérateur Equal, excepté que la commande Offset est toujours sensible à la casse, elle considère toujours les signes diacritiques, et elle n'est pas affectée par les instructions Considering et Ignoring.

Erreurs

Numéro	Message d'erreur
-50	Erreur de paramètre.
-108	Mémoire saturée.
-1700	Impossible de faire de <data> un item.
-1701	Le paramètre <name> est manquant pour <commandName>.
-1704	Certains paramètres sont invalides.
-1705	L'opération impliquant un élément liste a échoué.
-1708	<reference> ne comprend pas le message <commandName>.
-1718	La réponse n'est pas encore arrivée.

Open for Access

La commande Open for Access ouvre l'accès d'un fichier pour la lecture ou l'écriture. L'ouverture d'un fichier pour la lecture et l'écriture n'est pas la même chose que l'ouverture faite avec la commande Open. Le fichier est ouvert uniquement dans le sens où AppleScript y a accès pour la lecture et l'écriture de données ; le fichier n'apparaît pas dans une des fenêtres de l'application cible, et il ne doit même pas être un des fichiers de l'application cible.

Pour plus d'informations sur la lecture et l'écriture de données, voir "Utiliser les commandes Read / Write" (page 76).

Syntaxe

```
open for access referenceToFile [ write permission boolean ]
```

Paramètres

referenceToFile Une référence de la forme `file` *nameString* ou `alias` *nameString*. Si vous spécifiez un alias, le fichier doit déjà exister, car AppleScript doit localiser le fichier avant d'exécuter le script. Si vous spécifiez un fichier qui n'existe pas en utilisant la forme `file` *nameString*, la commande Open for Access crée un document SimpleText avec le nom et l'emplacement spécifiés, et ouvre son accès.
Classe : Reference

boolean Une expression qui s'évalue à `true` ou `false`. Si elle vaut `true`, AppleScript ouvre le fichier en lecture et en écriture. Si elle vaut `false` ou si le paramètre `write permission` est omis, AppleScript ouvre le fichier uniquement en lecture. Notez que la phrase `with write permission` est équivalente à `write permission true` ; de même, `without write permission` est équivalent à `write permission false`.
Classe : Boolean

Résultats

Un numéro de référence de fichier.

Exemples

Cet exemple ouvre le fichier nommé "Toto" à l'emplacement spécifié pour un prochain accès avec la commande Read. Le fichier sera créé s'il n'existe pas déjà.

```
tell application "SimpleText"
   open for access file "Disque Dur:Blagues:Toto"
end tell
```

Le prochain exemple ouvre le fichier spécifié avec le numéro de référence de fichier pour un accès ultérieur avec la commande Read ou Write. Le fichier sera créé s'il n'existe pas déjà.

```
tell application "AppleWorks"
   open for access file "Disque Dur:Blagues:Toto" ¬
      with write permission
end tell
```

Notes

Pour spécifier le nom (*nameString*) d'un fichier, utilisez une chaîne de caractères de la forme "*Disque Dur*:*Dossier1*:*Dossier2*. . . :*nomFichier*" comme il est décrit dans le tome 3 "Les objets et les références" du *Guide AppleScript version française*. Si vous indiquez uniquement le nom du fichier (*nomFichier*) au lieu du chemin en entier, AppleScript essaiera de trouver le fichier dans le répertoire courant.

Erreurs

Numéro	Message d'erreur
-35	Le disque <name> est introuvable..
-37	Mauvais nom de fichier.
-42	Trop de fichiers ouverts.
-43	Le fichier <name> est introuvable.
-44	Le disque <name> est protégé en écriture.
-49	Le fichier <name> est déjà ouvert.
-50	Erreur de paramètre.

Open Location

Définition rajoutée par rapport au guide Apple

La commande Open Location permet d'ouvrir un URL, dans votre application favorite par défaut de votre ordinateur. Souvent définie dans le tableau de bord Internet (config).

Syntaxe

```
open location string [ error reporting boolean ]
```

Paramètres

string	Une chaîne de caractères représentant l'adresse internet à ouvrir. *Classe* : String

boolean Une expression qui s'évalue à `true` ou `false`. Si elle vaut `true`, AppleScript affichera les erreurs de connexion. Si elle vaut `false` ou si le paramètre `error reporting` est omis, les erreurs de connexion ne seront pas affichées. Notez que la phrase `with error reporting` est équivalente à `error reporting true` ; de même, `without error reporting` est équivalent à `error reporting false`.
Classe : Boolean

Résultats

Aucun

Exemples

```
open location "trad.applescript.free.fr/" error reporting true
```

Notes

Lors de la première exécution de cette commande, une boîte de dialogue s'affichera et demandera confirmation de l'existence de paramètres réseau corrects. Une fois cette confirmation faite, la boîte de dialogue ne s'affichera plus pour les commandes Open Location ultérieures.

Path To

La commande Path To permet d'obtenir le chemin, sous la forme d'un alias ou d'une chaîne de caractères, de certains dossiers standards du disque de démarrage. Cette commande permet aussi d'obtenir l'emplacement sur le disque de l'application en avant-plan.

Syntaxe

`path to` *folderOrApplication* `[ as` *className* `]`

Paramètres

folderOrApplication	Une de ces constantes, mais la liste n'est pas complète : `me` (application active) `system folder` (Dossier Système) `font folder` (dossier "Polices") `control panels folder` (dossier "Tableaux de bord") `desktop` (Bureau) `extensions folder` (dossier "Extensions") `preferences folder` (dossier "Préférences") `apple menu items folder` (Dossier Menu Pomme) `startup disk` (Disque de démarrage) `temporary items` (dossier "Éléments temporaires") `trash` (Poubelle) `frontmost application` (application en avant-plan) `...`
className	L'identificateur de classe sous forme d'une chaîne. Si vous omettez ce paramètre, le chemin d'accès est retourné comme un alias.

Résultats

Un alias par défaut, ou une chaîne de caractères si vous indiquez `as string`.

Exemples

```
path to control panels folder
(* résultat : alias "Disque Dur:Dossier Système:Tableaux de
bord:" *)

tell application "AppleWorks"
   set x to path to it as string
end tell
-- résultat : "Disque Dur:Applications:AppleWorks:AppleWorks"

tell application "AppleWorks"
   activate
   tell application "SimpleText" to activate
   set x to path to frontmost application
end tell
return x
-- résultat : alias "Disque Dur:Applications:SimpleText"
```

Notes

Le paramètre optionnel as est utile si vous envoyez la commande Path To à une application distante. Si le chemin d'accès est retourné sous la forme d'une chaîne de caractères, vous pouvez utiliser la forme `file` *nameString* pour identifier le dossier ou l'application à travers le réseau, et actuellement la commande Path To n'essaiera pas de la localiser tant que vous n'exécuterez pas le script. Si le chemin d'accès est retourné sous la forme d'un alias et que vous l'utilisiez pour vous référer au dossier ou à l'application n'importe où dans le script, Path To essaiera alors de localiser le fichier chaque fois que vous modifierez le script, en requérant les paramètres appropriés d'accès et le mot de passe, s'il y a.

Erreurs

Numéro	Message d'erreur
-50	Erreur de paramètre.
-108	Mémoire saturée.
-1700	Impossible de faire de <data> un item.
-1701	Le paramètre <name> est manquant pour <commandName>.
-1704	Certains paramètres sont invalides.
-1705	L'opération impliquant un élément liste a échoué.
-1708	<reference> ne comprend pas le message <commandName>.
-1718	La réponse n'est pas encore arrivée.

Random Number

La commande Random Number génère un nombre aléatoire.

Syntaxe

```
random number                                  ¬
   [ numberToRandomize ]                       ¬
   [ from beginningNumber to endNumber ]       ¬
   [ with seed seedNumber ]
```

Paramètres

numberToRandomize Un nombre qui spécifie la limite haute de l'intervalle dans lequel vous voulez générer un nombre aléatoire. Si ce nombre est une valeur Real, la valeur retournée sera une valeur Real ; si ce nombre est une valeur Integer, la valeur retournée sera une valeur Integer.
Classe : Real ou Integer

beginningNumber Un nombre qui indique le début de l'intervalle dans lequel vous voulez générer un nombre aléatoire. Si ce nombre et *endNumber* sont tous les deux des Integers, la valeur retournée sera une valeur Integer. Si l'un des deux ou les deux sont des valeurs Real, la valeur retournée sera une valeur Real.
Classe : Real ou Integer

endNumber Un nombre qui indique la fin de l'intervalle dans lequel vous voulez générer un nombre aléatoire. Si ce nombre et *beginningNumber* sont tous les deux des Integers, la valeur retournée sera une valeur Integer. Si l'un des deux ou les deux sont des valeurs Real, la valeur retournée sera une valeur Real.
Classe : Real ou Integer

seedNumber Un nombre qui indique le nombre à utiliser comme référence pour la génération d'un nombre aléatoire dans une séquence répétitive.
Classe : Real ou Integer

Résultats

Un nombre aléatoire dans les limites spécifiées. Si aucun paramètre n'est renseigné, Random Number retourne une valeur comprise entre 0.0 et 1.0.

Exemples

```
display dialog ¬
   "Un nombre aléatoire entre 0 et 1 : " & ¬
   (random number)

display dialog ¬
   "Un nombre entier aléatoire entre 1 et 10 : " & ¬
   (random number from 1 to 10)
```

```
display dialog ¬
   "Un nombre réel aléatoire entre 1 et 10.0 : " & ¬
   (random number from 1 to 10.0)

display dialog ¬
   "Un nombre réel aléatoire entre -10.0 et 10 : " & ¬
   (random number from -10.0 et 10)

display dialog ¬
   "Un nombre entier aléatoire entre 1 et 10, " & ¬
   "12 est la référence : " & ¬
   (random number from 1 to 10 with seed 12)
```

Après que la référence soit réglée, les nombres générés ultérieurement par la commande Random Number, dans le même script, peuvent être prévus à l'avance.

```
display dialog ¬
   "Cela devrait être 10 : " & ¬
   (random number from 1 to 10)
display dialog ¬
   "Cela devrait être 9 : " & ¬
   (random number from 1 to 10)
```

Remettre la référence à 0 (`with seed 0`) provoque de nouveau la génération d'un véritable nombre aléatoire chaque fois que la commande est appelée.

```
display dialog ¬
   "Après la réinitialisation, un vrai nombre aléatoire : " & ¬
   (random number from 1 to 10 with seed 0)
```

Erreurs

Numéro	Message d'erreur
-50	Erreur de paramètre.
-108	Mémoire saturée.
-1700	Impossible de faire de <data> un item.
-1701	Le paramètre <name> est manquant pour <commandName>.
-1704	Certains paramètres sont invalides.

-1705	L'opération impliquant un élément liste a échoué.
-1708	<reference> ne comprend pas le message <commandName>.
-1718	La réponse n'est pas encore arrivée.

Read

La commande Read lit les données d'un fichier, en commençant depuis la marque du fichier jusqu'à la fin du fichier.

Pour plus d'informations sur la commande Read, voir "Utiliser les commandes Read / Write" (page 76).

Syntaxe

```
read referenceToFile                                          ¬
   [ from startByte ]                                         ¬
   [ for bytesToRead | to byteToReadTo                        ¬
       | until delimiterIncluded | before delimiterExcluded ] ¬
   [ as className [ using delimiter[s] delimiters ] ]
```

Paramètres

referenceToFile	Une référence de la forme `file` *nameString* ou `alias` *nameString*, ou un numéro de référence de fichier obtenu précédemment avec la commande Open for Access (voir "Notes"). *Classe :* Reference ou Integer
startingByte	La position de l'octet à partir duquel commence la lecture. Un nombre entier positif indique la position par rapport au début du fichier, un nombre entier négatif indique la position par rapport à la fin du fichier. *Classe :* Integer
bytesToRead	Le nombre d'octets à lire. Si le paramètre `from` *startingByte* est renseigné, la commande Read lira *bytesToRead* octets en commençant au point de départ spécifié ; sinon, la commande Read débutera la lecture à la marque du fichier. Si la valeur de ce paramètre est négative, une erreur survient. *Classe :* Integer

byteToReadTo La position de l'octet pour lire jusqu'à. Si le paramètre `from` *startingByte* est renseigné, la commande Read lira à partir du point de départ spécifié jusqu'à *byteToReadTo* ; sinon, la commande Read débutera la lecture à la marque du fichier. Un nombre entier positif indique la position par rapport au début du fichier, un nombre entier négatif indique la position par rapport à la fin du fichier.
Classe : Integer

delimiterIncluded Un délimiteur (comme tab ou return) marquant la fin de la lecture. Le délimiteur spécifié est compris dans la lecture (à moins qu'il soit un délimiteur de fin de fichier, lequel n'est pas compris). Si le paramètre from *startingByte* est renseigné, la commande Read lira à partir du point de départ spécifié jusqu'au délimiteur spécifié ; sinon, la commande Read commencera la lecture à la marque du fichier.
Classe : String

delimiterExcluded Un délimiteur (comme tab ou return) marquant la fin de la lecture. Le délimiteur n'est pas compris dans la lecture. Si le paramètre from *startingByte* est renseigné, la commande Read lira à partir du point de départ spécifié jusqu'au délimiteaur spécifié ; sinon, la commande Read commencera la lecture à la marque du fichier.
Classe : String

className La classe de la donnée à lire. La commande Read lit le nombre d'octets approprié pour une valeur de la classe spécifiée par ce paramètre. Pour plus de détails, voir "Notes"
Classe : Class

delimiters Si les données devant être lues sont au format texte, vous pouvez utiliser ce paramètre pour spécifier les délimiteurs que la commande Read devra utiliser lorsqu'elle interprète les données comme des valeurs de la classe spécifiée par *className.* Pour plus de détails, voir "Notes".
Classe : String ou Constant, ou une liste à deux éléments de la classe String ou Constant.

Résultats

Si la commande Read est un succès, elle retourne les données lues comme du texte (à moins que le paramètre `as` soit renseigné).

Exemples

L'exemple suivant lit `monFichier` du 12ème octet jusqu'à la fin du fichier.

```
read file "Disque Dur:monFichier" from 12
```

L'exemple suivant lit `monFichier` du 12ème octet en partant de la fin jusqu'à la fin du fichier.

```
read file "Disque Dur:monFichier" from -12
```

L'exemple suivant lit 24 octets de `monFichier` en démarrant au 12ème octet. Si la fin du fichier est atteinte avant que les 24 octets aient été lus, une erreur est retournée.

```
read file "Disque Dur:monFichier" from 12 for 24
```

L'exemple suivant lit `monFichier` en démarrant à la fin du fichier et en lisant à l'envers jusqu'au 3ème octet à partir de la fin.

```
read file "Disque Dur:monFichier" from -1 to -3
```

Si les derniers caractères du fichier `monFichier` étaient `"123456"`, l'instruction précédente retournerait `"654"`.

Notes

La marque de fichier est un marqueur utilisé par le File Manager indiquant l'octet à partir duquel la commande Read s'attend à commencer la lecture des données. Par défaut, la marque de fichier est positionnée sur le premier octet du fichier. Toutefois, exécuter un script comme celui qui suit provoque le déplacement de la marque de fichier de `monFichier` :

```
read file "Disque Dur:monFichier" from 1 to 4
```

La marque de fichier pour `monFichier` est maintenant positionnée sur l'octet 5, donc la prochaine commande Read dans le même script débutera à l'octet 5. Par exemple, la commande

```
read file "Disque Dur:monFichier" for 4
```

lira les octets 5 à 8.

Pour spécifier le nom (*nameString*) d'un fichier, utilisez une chaîne de caractères de la forme "*Disque Dur*:*Dossier1*:*Dossier2*. . . :*nomFichier*" comme il est décrit dans le tome 3 "Les objets et les références" du *Guide AppleScript version française*. Si vous indiquez uniquement le nom du fichier (*nomFichier*) au lieu du chemin en entier, AppleScript essaiera de trouver le fichier dans le répertoire courant.

Si vous spécifiez une référence à un fichier ou à un alias, la commande Read essaiera de faire correspondre la référence avec un fichier précédemment ouvert avec la commande Open for Access. Si une correspondance est trouvée, elle lira simplement les données spécifiées. Si aucune correspondance n'est trouvée mais que le fichier peut être localisé sur le disque, la commande Read ouvrira le fichier, lira les données spécifiées, puis fermera le fichier. La marque de fichier, pour un fichier ouvert de cette manière, est toujours positionnée au début du fichier. Si le fichier ne peut pas être du tout localisé, l'application recevant la commande Read retournera une erreur numéro -43.

Si vous spécifiez un numéro de référence de fichier précédemment obtenu avec la commande Open for Access, la commande Read lira immédiatement les données spécifiées.

Vous pouvez utiliser le paramètre as *className* pour spécifier comment la commande Read devra interpréter les données à lire. Si les données devant être lues ne sont pas valides pour la classe de valeur spécifiée, la commande Read retournera une erreur.

La suite de cette section décrira quelques une des classes de valeurs pouvant être spécifiées et la nature des données retournées si la commande Read lit les données avec succès.

`as list` La commande Read retournera une liste uniquement si les données devant être lues sont écrites comme une liste AppleScript. Si les données devant être lues sont un texte délimité, vous pouvez spécifier les délimiteurs utilisés dans les données avec le paramètre `using delimiter`, la commande Read créera alors une liste AppleScript basée sur ces délimiteurs.

Par exemple, le script suivant retourne une liste à partir de `monFichier` en utilisant ensemble les caractères tab et return pour séparer chaque élément de la liste :

```
read file "Disque Dur:monFichier" as {text} ¬
      using delimiters {return, tab}
```

La liste obtenue, comme n'importe quelle autre liste AppleScript, est délimitée par des virgules. Vous ne pouvez pas spécifier plus que deux délimiteurs ; si vous le faites, la commande Read retournera l'erreur numéro -50.

`as record` La commande Read retournera un enregistrement uniquement si les données devant être lues sont écrites comme un enregistrement AppleScript. La commande Read ne peut pas contraindre d'autres valeurs en Record.

➥ **Note des traducteurs francophones (source Liste AS-Fr)**
Mais la classe de valeur Record n'est fonctionnelle que depuis Mac OS 9.2.1 / AS 1.6. Dans les versions inférieures, c'est bogué. ●

`as integer` Si les données se composent d'un unique nombre entier, la commande Read retournera le nombre entier. Si les données se composent de plusieurs nombres entiers, la commande Read retournera une liste de nombres entiers.

`as text` La commande Read retournera les données sous la forme d'une chaîne de caractères. C'est le comportement par défaut si le paramètre `as` *className* est omis.

`as real` Si les données se composent d'un unique nombre réel, la commande Read retournera le nombre réel. Si les données se composent de plusieurs nombres réels, la commande Read retournera une liste de nombres réels.

`as short` La valeur short est définie par le complément de pilotage sur 2-octets. Cela peut être utile si vous lisez des données depuis un fichier qui utilise des nombres entiers courts, plutôt que des nombres entiers longs, encodés sur 4 octets, comme dans AppleScript. La commande Read interprètera les données comme une ou plusieurs valeurs encodées sur 2-octets. Si les données se composent de plusieurs valeurs courtes, AppleScript retournera une liste de nombres courts. Si les données sont du texte, vous pouvez spécifier les délimiteurs utilisés dans les données avec le paramètre `using delimiter`, la commande Read essaiera de contraindre chaque élément séparé par les délimiteurs en une valeur courte.

`as boolean` Si les données se composent d'une valeur booléenne codée sur 1-octet, la commande Read retournera la valeur booléenne. Si les données se composent de plusieurs valeurs booléennes, la commande Read retournera une liste de valeurs booléennes.

`as data` La commande Read retournera les données sous la forme d'une suite continue d'octets hexadécimaux non-interprétés.

Vous pouvez aussi spécifier d'autres types en indiquant le code approprié sur 4 caractères encadré par des guillemets ("") comme dans l'exemple qui suit :

```
read file "Disque Dur:monFichier" as "PICT"
-- retournera des données du type ‘PICT’
```

Erreurs

Numéro	Message d'erreur
-38	Le fichier <name> n'a pas été ouvert.
-39	Erreur de fin de fichier.
-50	Erreur de paramètre.
-51	Erreur de numéro de référence de fichier.
-108	Mémoire saturée.
-1700	Impossible de faire de <data> un item.
-1701	Le paramètre <name> est manquant pour <commandName>.

-1704	Certains paramètres sont invalides.
-1705	L'opération impliquant un élément liste a échoué.
-1715	Certains paramètres n'ont pas été compris.
-1718	La réponse n'est pas encore arrivée.

Round

La commande Round arrondit ou tronque un nombre et retourne un nombre entier.

Par défaut, la commande Round arrondit au nombre entier le plus proche. Mais vous pouvez aussi rajouter des paramètres optionnels, comme `rounding up` qui arrondit au nombre entier immédiatement supérieur

Syntaxe

```
round number ¬
   [ rounding ( up | down | toward zero | ¬
      to nearest | as taught in school (OS 9.1/AS 1.5.5)) ]
```

Paramètres

number	Le nombre à arrondir. *Classe* : Number

Résultats

Le résultat est un nombre entier : la valeur arrondie.

Exemples

```
round -3.67
-- résultat : -4
round 3.67
-- résultat :4

round -3.67 rounding up
-- résultat :-3
round 3.67 rounding up
```

```
-- résultat : 4

round -3.67 rounding down
-- résultat :-4
round 3.67 rounding down
-- résultat :3

round -3.67 rounding toward zero
-- résultat :-3
round 3.67 rounding toward zero
-- résultat :3

round -3.67 rounding to nearest
-- résultat :-4
round 3.67 rounding to nearest
-- résultat :4

-- apparu avec Mac OS 9.1/AS 1.5.5
round 3.4 rounding as taught in school
-- résultat : 3
round 3.6 rounding as taught in school
-- résultat : 4
```

Erreurs

Numéro	Message d'erreur
-50	Erreur de paramètre.
-108	Mémoire saturée.
-1700	Impossible de faire de <data> un item.
-1701	Le paramètre <name> est manquant pour <commandName>.
-1704	Certains paramètres sont invalides.
-1705	L'opération impliquant un élément liste a échoué.
-1708	<reference> ne comprend pas le message <commandName>.
-1718	La réponse n'est pas encore arrivée.

Run Script

La commande Run Script exécute un script qui est, soit un script dans le même script, soit un fichier script extérieur.

Syntaxe

```
run script referenceOrString                ¬
   [ with parameters listOfParameters ]     ¬
   [ in scriptingComponent ]
```

Paramètres

referenceOrString	Une référence telle que `file` *nameString* ou `alias` *nameString* qui spécifie un fichier script, ou une chaîne de caractères composée d'un script valide. *Classe* : Reference ou String
listOfParameters	Une liste de paramètres devant être transmis au gestionnaire Run de la cible. *Classe* : List
scriptingComponent	Le nom du scripting component à utiliser lors de l'exécution du script. *Classe* : String

Résultats

La valeur retournée par l'exécution du script.

Exemples

L'exemple suivant exécute le script `beep 3`, déclenchant trois fois le son d'alerte.

```
run script "beep 3"
```

L'exemple suivant s'assure que le script `beep 3` est bien exécuté par le scripting component AppleScript.

```
run script "beep 3" in "AppleScript"
```

L'exemple suivant exécute le gestionnaire Run du fichier script appelé `monScript`.

```
run script file "Disque Dur:Scripts:monScript"
```

Notes

Pour spécifier le nom (*nameString*) d'un fichier, utilisez une chaîne de caractères de la forme "*Disque Dur*:*Dossier1*:*Dossier2*. . . :*nomFichier*" comme il est décrit dans le tome 3 "Les objets et les références" du *Guide AppleScript version française*. Si vous indiquez uniquement le nom du fichier (*nomFichier*) au lieu du chemin en entier, AppleScript essaiera de trouver le fichier dans le répertoire courant.

Erreurs

Numéro	Message d'erreur
-50	Erreur de paramètre.
-108	Mémoire saturée.
-192	Une ressource est introuvable.
-1700	Impossible de faire de <data> un item.
-1701	Le paramètre <name> est manquant pour <commandName>.
-1704	Certains paramètres sont invalides.
-1705	L'opération impliquant un élément liste a échoué.
-1708	<reference> ne comprend pas le message <commandName>.
-1718	La réponse n'est pas encore arrivée.
-1750	Erreur du système descript.
-1751	Numéro d'identification de script invalide.
-1753	Erreur de script.

Scripting Components

La commande Scripting Components retourne la liste de tous les langages de pilotage disponibles pour l'application cible.

Syntaxe

```
scripting components
```

Résultats

Une liste de chaînes de caractères.

Exemples

```
scripting components
-- résultat : {"JavaScript","AppleScript"}
```

Notes

Un composant de pilotage (scripting component) est un logiciel, comme AppleScript, qui supporte l'OSA (Open Scripting Architecture). L'OSA fournit un mécanisme, basé sur les Apple Events, qui permet aux utilisateurs de contrôler plusieurs applications, au moyen de scripts écrits parmi une variété de langages de pilotage. Chaque langage de pilotage correspond à un unique composant de pilotage. Par contre, un seul langage de pilotage peut inclure différents dialectes.

Erreurs

Numéro	Message d'erreur
-108	Mémoire saturée.

Set The Clipboard To

Définition rajoutée par rapport au guide Apple

La commande Set The Clipboard To règle le contenu du presse-papiers de l'application active.

Syntaxe

`Set The Clipboard to` *data*

Paramètres

data	Les données à mettre dans le presse-papiers. *Classe :* n'importe

Résultats

Aucun.

Exemples

```
set the clipboard to "Toto"
-- la chaîne "Toto" est stockée dans le presse-papiers

set the clipboard to 123
-- le nombre 123 est stocké dans le presse-papiers
```

Notes

La commande Set The Clipboard To règle le contenu du presse-papiers de l'application active. Si plus d'une application est ouverte, utilisez la commande Activate pour mettre l'application désirée au premier-plan.

La commande Set The Clipboard to peut gérer différents types de données. Le texte stylé est le type de données par défaut, le texte standard est utilisé comme texte de sauvegarde. Les caractères internationaux, les images et les sons sont également gérés.

Il est possible d'utiliser la classe de valeur Record pour le paramètre *typeClass* (`the clipboard as record`), afin d'obtenir toutes les données internes du presse-papiers sous forme d'un enregistrement. Par contre, si vous stockez un enregistrement dans le presse-papiers, il n'y a pas blocage, mais vous ne pourrez pas récupérer, par la suite, les données sous la forme de l'enregistrement initial.

```
set the clipboard to {nom:"aaaa",prenom:"bbbb"}
the clipboard as record
-- résultat : {list:{}}
the clipboard
-- résultat : {"nom","Dugenou","prenom","Toto"}
```

Cette particularité ne devrait pas être un problème, car mettre un

enregistrement dans le presse-papiers ne semble pas vraiment utile.

Set EOF

La commande Set EOF sert à régler la position de fin de fichier du fichier spécifié. Pour plus d'informations, voir "Utiliser les commandes Read / Write" (page 76).

Syntaxe

```
set eof referenceToFile to integer
```

Paramètres

referenceToFile	Une référence telle que `file` *nameString* ou `alias` *nameString*, ou un numéro de référence de fichier obtenu précédemment avec la commande Open for Access. *Classe* : Reference ou Integer
integer	Le nombre d'octets réglant la position de fin de fichier. *Classe* : Integer

Résultats

Aucun

Exemples

```
set eof file "Disque Dur:monFichier" to 10
```

Notes

Pour spécifier le nom (*nameString*) d'un fichier, utilisez une chaîne de caractères de la forme "*Disque Dur*:*Dossier1*:*Dossier2*. . . :*nomFichier*" comme il est décrit dans le tome 3 "Les objets et les références" du *Guide AppleScript version française*. Si vous indiquez uniquement le nom du fichier (*nomFichier*) au lieu du chemin en entier, AppleScript essaiera de trouver le fichier dans le répertoire courant.

Si vous spécifiez une référence à un fichier ou à un alias, la commande Set EOF essaiera de faire correspondre la référence avec un fichier précédemment ouvert avec la commande Open for Access (avec autorisation d'écriture). Si le fichier n'a été ouvert que pour la lecture, la commande Set EOF retournera une erreur numéro -61. Si une correspondance est trouvée, Set EOF règlera la position de fin de fichier comme il est spécifié. Si aucune correspondance n'est trouvée, Set EOF ouvrira le fichier, règlera la position de fin de fichier, puis le refermera.

Si vous spécifiez un numéro de référence de fichier précédemment obtenu avec la commande Open for Access, la commande Set EOF règlera immédiatement la position de fin de fichier.

Important
Si le fichier est plus long que la position de fin de fichier réglée avec la commande Set EOF, le fichier sera tronqué au niveau du réglage et les données correspondant à la partie excédentaire seront définitivement perdues. Si le fichier est plus petit que la position de fin de fichier réglée avec la commande Set EOF, le fichier sera agrandi pour correspondre au réglage, mais les données supplémentaires seront sans signification. ▲

Erreurs

Numéro	Message d'erreur
-34	Le disque <name> est saturé.
-38	Le fichier <name> n'a pas été ouvert.
-44	Le disque <name> est protégé en écriture.
-45	Le fichier <name> est verrouillé.
-46	Le disque <name> est verrouillé.
-50	Erreur de paramètre.
-51	Erreur de numéro de référence de fichier.
-61	Fichier non ouvert avec autorisation en écriture.

Set Volume

Définition rajoutée par rapport au guide Apple

La commande Set Volume permet de régler le volume sonore de votre ordinateur.

Syntaxe

```
set volume integer
```

Paramètres

integer Un nombre entier compris entre 0 (silence) et 7 (maximum).
Classe : Integer

Résultats

Aucun

Exemples

```
set volume 0 -- pas de son

set volume 7 -- volume à fond
```

Start Log

La commande Start Log réactive l'affichage des Events (elle désactive la commande Stop Log). Start Log est une commande propriétaire pour l'Éditeur de scripts d'Apple. Elle fait partie, soit du dialecte anglais, soit de l'extension AppleScript, suivant la version de Mac OS et d'AS.

Si les cases "Afficher les événements" et "Afficher les résultats d'événements" de la fenêtre Session d'état de l'Éditeur de scripts sont cochées, les commandes envoyées par l'Éditeur de scripts et les résultats retournés par chaque Event sont affichés dans cette fenêtre. Le choix des éléments à afficher se fait en cochant les cases correspondantes. Vous pouvez aussi sauvegarder une copie du texte de la fenêtre Session d'état en

choisissant “Enregistrer sous …” du menu Fichier.

Syntaxe

```
start log
```

Résultats

Aucun

Exemples

L’exemple suivant montre comment démarrer l’affichage et son fonctionnement. Pour voir la commande Start Log fonctionner, vous devez au préalable ouvrir la fenêtre Session d’état et vérifier que les deux cases sont bien toutes les deux cochées.

```
start log
display dialog "Bonjour"
```

Après avoir exécuté ce script, l’enregistrement des Events envoyés et les résultats retournés apparaissent dans la fenêtre Session d’état. Comme la commande Display Dialog, dans ce script, n’était pas à l’intérieur d’une instruction Tell, l’application Éditeur de script l’a envoyée à l’application courante - ici, à elle-même. La fenêtre Session d’état montre alors ensemble l’instruction Tell implicite et le résultat retourné par la commande Display Dialog.

Erreurs

Numéro	Message d’erreur
-50	Erreur de paramètre.
-108	Mémoire saturée.
-1700	Impossible de faire de <data> un item.
-1701	Le paramètre <name> est manquant pour <commandName>.
-1704	Certains paramètres sont invalides.

-1705	L'opération impliquant un élément liste a échoué.
-1708	<reference> ne comprend pas le message <commandName>.
-1718	La réponse n'est pas encore arrivée.

Stop Log

La commande Stop Log arrête l'affichage des Events. Stop Log est une commande propriétaire pour l'Éditeur de scripts d'Apple. Elle fait partie, soit du dialecte anglais, soit de l'extension AppleScript, suivant la version de Mac OS et d'AS.

Pour plus d'informations, voir la commande précédente Start Log.

Syntaxe

```
stop log
```

Résultats

Aucun

Exemples

```
stop log
```

Notes

Notez que la commande Stop Log n'a aucune influence sur la commande Log, les valeurs des variables sont reportées dans la fenêtre Session d'état même après une instruction Stop Log. De même, la commande Log n'a pas d'influence sur la commande Stop Log, l'affichage des Events reste suspendu pour les Events non spécifiés par l'instruction Log.

Erreurs

Numéro	Message d'erreur
-50	Erreur de paramètre.
-108	Mémoire saturée.

-1700	Impossible de faire de <data> un item.
-1701	Le paramètre <name> est manquant pour <commandName>.
-1704	Certains paramètres sont invalides.
-1705	L'opération impliquant un élément liste a échoué.
-1708	<reference> ne comprend pas le message <commandName>.
-1718	La réponse n'est pas encore arrivée.

Store Script

La commande Store Script stocke un script-objet spécifié dans un fichier spécifié.

Syntaxe

```
store script scriptObjectVariable          ¬
   [ in referenceToFile ]                  ¬
   [ replacing replacementOption ]
```

Paramètres

scriptObjectVariable	Le nom d'un script-objet déclaré précédemment dans le même script. *Classe* : Script
referenceToFile	Une référence telle que `file` *nameString* ou `alias` *nameString* (voir "Notes). Si ce paramètre est omis, la commande Store Script affichera une boîte de dialogue et demandera à l'utilisateur de choisir un fichier dans lequel enregistrer le script. *Classe* : Reference
replacementOption	Une de ces constantes : `ask`, `yes` ou `no`. La constante `ask` provoque l'affichage d'une boîte de dialogue dans laquelle l'utilisateur doit choisir entre, écraser le contenu du fichier spécifié par *scriptObjectVariable*, renommer le fichier ou annuler l'opération. Si le paramètre `replacing` est omis, la boîte de la dialogue sera par défaut affichée. La constante `yes` indique que la commande Store Script doit écraser le fichier, `no` qu'elle ne doit pas l'écraser s'il existe déjà. *Classe* : String

Résultats

Aucun

Exemples

Le script-objet suivant est utilisé par la suite dans les exemples :

```
script demoStore
   property : Red : 127
   property Green : 128
   property Blue : 127
   on returnRGB()
       return Red & Green & Blue
   end return
end script
```

Dans l'exemple suivant le script-objet `demoStore` est stocké dans un fichier appelé Store my RGB :

```
store script demoStore in file ¬
   "Disque Dur:Store my RGB" replacing yes
```

Le paramètre `replacing yes` indique que la commande Store Script devra écraser le fichier existant.

Quelques instructions plus tard, vous voulez accéder à `demoStore` afin de modifier une de ses propriété. Vous devrez tout d'abord charger le script-objet `demoStore`, modifier la propriété et le restocker de nouveau sur le disque, comme dans le script suivant :

```
set objRef to load script file "Disque Dur:Store my RGB"
set objRef's Red to 250
store script objRef in file "Disque Dur:Store my RGB"
```

Dans ce cas, le paramètre `replacing` est omis, donc la commande Store Script affichera une boîte de dialogue demandant à l'utilisateur de choisir entre, écraser le fichier, le renommer ou annuler l'opération.

Notes

Pour spécifier le nom (*nameString*) d'un fichier, utilisez une chaîne de caractères de la forme "*Disque Dur*:*Dossier1*:*Dossier2*...:*nomFichier*" comme

il est décrit dans le tome 3 “Les objets et les références” du *Guide AppleScript version française*. Si vous indiquez uniquement le nom du fichier (*nomFichier*) au lieu du chemin en entier, AppleScript essaiera de trouver le fichier dans le répertoire courant.

Erreurs

Numéro	Message d’erreur
-108	Mémoire saturée.
-128	Annulé par l’utilisateur.
-192	Une ressource est introuvable.
-1700	Impossible de faire de <data> un item.
-1701	Le paramètre <name> est manquant pour <commandName>.
-1704	Certains paramètres sont invalides.
-1705	L’opération impliquant un élément liste a échoué.
-1708	<reference> ne comprend pas le message <commandName>.
-1718	La réponse n’est pas encore arrivée.

The Clipboard

Définition rajoutée par rapport au guide Apple

La commande The Clipboard retourne le contenu du presse-papiers de l’application active.

Syntaxe

```
the clipboard [ as typeClass ]
```

Paramètres

typeClass — La classe de valeur des données. Ce paramètre permet de ne récupérer que les données correspondant à la classe spécifiée (voir “Notes” pour la classe Record). Si ce paramètre est omis, la commande The Clipboard retourne le contenu du presse-papiers en conservant le format d’origine.
Classe : Class

Résultats

Le contenu du presse-papiers.

Exemples

```
the clipboard as string
```

Notes

La commande The Clipboard obtient le contenu du presse-papiers de l'application active. Si plus d'une application est ouverte, utilisez la commande Activate pour mettre l'application désirée au premier-plan.

Habituellement, le terme "the" est optionnel dans AppleScript. Cependant, certaines applications comportent déjà dans leur dictionnaire une propriété Clipboard. Si tel est le cas, employez Clipboard (seul) pour la propriété de l'application et The Clipboard pour la commande du Compléments standard.

La commande The Clipboard peut gérer différents types de données. Le texte stylé est le type de données par défaut, le texte standard est utilisé comme texte de sauvegarde. Les caractères internationaux sont également gérés, les images et les sons aussi.

Il est possible d'utiliser la classe de valeur Record pour le paramètre *typeClass* (`the clipboard as record`), afin d'obtenir toutes les données internes du presse-papiers sous forme d'un enregistrement. Par contre, si vous stockez un enregistrement dans le presse-papiers, il n'y a pas blocage, mais vous ne pourrez pas récupérer, par la suite, les données sous la forme de l'enregistrement initial.

```
set the clipboard to {nom:"aaaa",prenom:"bbbb"}
the clipboard as record
-- résultat : {list:{}}
the clipboard
-- résultat : {"nom","Dugenou","prenom","Toto"}
```

Cette particularité ne devrait pas être un problème, car mettre un enregistrement dans le presse-papiers ne semble pas vraiment utile.

Time to GMT

La commande Time to GMT retourne la différence, en secondes, entre l'heure courante et celle du méridien de Greenwich (GMT).

Syntaxe

```
time to GMT
```

Résultats

Un nombre entier indiquant la différence en secondes entre l'heure courante et celle GMT.

Exemples

La commande Time to GMT retourne la différence, en secondes, entre l'heure de votre ordinateur et celle GMT. Par exemple, si l'heure de votre ordinateur est réglée sur le fuseau horaire de Paris, la commande Time to GMT retournera ce résultat :

```
time to GMT
-- résultat : 7200
```

Vous pourrez alors utiliser cette valeur pour écrire un script qui vous donnera le décalage horaire entre l'heure de votre ordinateur et celle de n'importe quelle zone du globe.

```
prop offsetParis : -7200

set x to ((time to GMT) ÷ 60) ÷ 60
set y to x - (((offsetParis) ÷ 60) ÷ 60)

if y = 0 then
   display dialog "Même fuseau horaire qu'à Paris"
else
   if y > 0 then
      display dialog ¬
         (y & " heures de plus avec Paris") as string
   else
      display dialog ¬
         (-y & " heures de moins avec Paris") as string)
```

```
   end if
end if
```

Pour constater les effets de ce script, essayer en réglant l'heure de votre ordinateur depuis le tableur de bord Planisphère (ou Date et Heure pour OS ≥ 8.5) et en choisissant différents points du globe, comme Londres, Tokyo ou New-York.

Erreurs

Numéro	Message d'erreur
-108	Mémoire saturée.
-1700	Impossible de faire de <data> un item.
-1701	Le paramètre <name> est manquant pour <commandName>.
-1704	Certains paramètres sont invalides.
-1705	L'opération impliquant un élément liste a échoué.
-1708	<reference> ne comprend pas le message <commandName>.
-1718	La réponse n'est pas encore arrivée.

Write

La commande Write écrit des données dans un fichier spécifié, en commençant à la marque de fichier.

Pour plus d'informations sur cette commande, voir "Utiliser les commandes Read / Write" (page 76).

Syntaxe

```
write dataToWrite to referenceToFile       ¬
   [ for bytesToWrite ]                    ¬
   [ starting at startingByte ]
```

Paramètres

dataToWrite — Les données qui doivent être écrites. Le format des données doit correspondre au type du fichier.
Classe : varie suivant les applications

referenceToFile — Une référence de la forme `file` *nameString* ou `alias` *nameString*, ou un numéro de référence de fichier obtenu précédemment par la commande Open for Access (voir “Notes”).
Classe : Reference ou Integer

bytesToWrite — Le nombre d’octets à écrire. La commande Write retournera une erreur si vous indiquez un nombre négatif pour ce paramètre.
Classe : Integer

startingByte — La position de l’octet à partir duquel commence l’écriture. Un nombre entier positif indique la position par rapport au début du fichier, et un nombre entier négatif indique la position par rapport à la fin du fichier.
Classe : Integer

➥ **Note des traducteurs francophones**
Une fonction très utilisée, mais non documentée, est d’utiliser `eof` (≠ `get eof` !!!!) pour *startingByte* :

```
write data toWrite to reference toFile starting at eof
```

Cela permet d’écrire directement en fin de fichier. ●

Résultats

Aucun

Exemples

L’exemple suivant écrit "abcde" dans le fichier monFichier.

```
write "abcde" to file "Disque Dur:monFichier"
```

L’exemple suivant retourne une erreur car moins d’octets sont spécifiés pour le paramètre *dataToWrite* que pour le paramètre *bytesToWrite* :

```
write "abcde" to file "Disque Dur:monFichier" for 8
```

Si les données à écrire sont plus longues que le nombre d'octets spécifié par le paramètre *bytesToWrite*, la commande Write tronquera les données. Par exemple, la commande Write suivante écrit le nombre 5 dans le fichier monFichier comme un entier court (sur 2 octets) plutôt que comme un entier long (sur 4 octets) :

```
write 5 to file "Disque Dur:monFichier" for 2
```

L'exemple suivant indique une valeur négative pour le paramètre *startingByte*. Le script écrit le nombre 5 comme un entier court à partir 8ème octet en partant de la fin du fichier :

```
write 5 to file "Disque Dur:monFichier" starting at -8 for 2
```

Notes

La marque de fichier est un marqueur utilisé par le File Manager indiquant l'octet à partir duquel la commande Read s'attend à commencer la lecture des données. Par défaut, la marque de fichier est positionnée sur le premier octet du fichier. La commande Write commence d'écrire à la marque de fichier courante et règle la position de la marque de fichier sur l'octet suivant immédiatement le dernier octet écrit. La commande Read peut aussi réinitialiser la position de la marque de fichier.

Pour régler la position de la marque de fichier sans lire ou écrire de données, écrivez une chaîne de caractères de longueur zéro à l'octet pour lequel vous voulez régler la position de la marque de fichier. Par exemple, le script suivant règle la position de la marque de fichier du fichier `fileRefNum` sur le 4ème octet du fichier :

```
write "" to fileRefNum starting at 4
```

Pour spécifier le nom (*nameString*) d'un fichier, utilisez une chaîne de caractères de la forme "*Disque Dur*:*Dossier1*:*Dossier2*. . . :*nomFichier*" comme il est décrit dans le tome 3 "Les objets et les références" du *Guide AppleScript version française*. Si vous indiquez uniquement le nom du fichier (*nomFichier*) au lieu du chemin en entier, AppleScript essaiera de trouver le fichier dans le répertoire courant.

Si vous spécifiez une référence à un fichier ou à un alias, la commande Write essaiera de faire correspondre la référence avec un fichier précédemment ouvert avec la commande Open for Access. Si une correspondance est trouvée, elle écrira simplement les données spécifiées. Si aucune correspondance n'est trouvée mais que le fichier peut être localisé sur le disque, la commande Write ouvrira le fichier, lira les données spécifiées, puis fermera le fichier. La marque de fichier, pour un fichier ouvert de cette manière, est toujours positionnée au début du fichier. Si le fichier ne peut pas du tout être localisé, l'application recevant la commande Write retournera une erreur numéro -43.

Si vous spécifiez un numéro de référence de fichier précédemment obtenu avec la commande Open for Access, la commande Write écrira immédiatement les données spécifiées.

Erreurs

Numéro	Message d'erreur
-34	Le disque <name> est saturé.
-38	Le fichier <name> n'a pas été ouvert.
-44	Le disque <name> est protégé en écriture.
-45	Le fichier <name> est verrouillé.
-49	Le fichier <name> est déjà ouvert.
-50	Erreur de paramètre.
-51	Erreur de numéro de référence de fichier.
-61	Fichier non ouvert avec autorisation en écriture.
-108	Mémoire saturée.

Chapitre 5

Utiliser les commandes Read/Write

Les commandes Get EOF, Set EOF, Read, Write, Open for Access et Close Access permettent d'ouvrir l'accès d'un fichier, d'obtenir et de régler sa taille, de lire les données contenues dans le fichier, d'insérer de nouvelles données et de fermer son accès. Ces commandes permettent d'utiliser, depuis un script, certaines capacités du File Manager, la partie de Mac OS qui contrôlent les fichiers.

Attention
Ces commandes doivent être utilisées avec précaution. Utiliser incorrectement ces commandes peut entraîner des pertes de données irréversibles.▲

La plupart de ces commandes permettent de spécifier un numéro de référence de fichier au lieu d'une référence à un fichier. Un numéro de référence de fichier est un nombre entier, assigné par le File Manager, qui identifie de façon unique un fichier. Vous pouvez obtenir un numéro de référence de fichier avec la commande Open for Access, alors vous utiliserez le nombre retourné pour vous référer au même fichier jusqu'à ce que vous utilisiez la commande Close Access pour fermer l'accès du fichier. Il est généralement préférable d'utiliser un numéro de référence de fichier plutôt qu'une référence à un fichier, car la localisation du fichier est plus rapide.

Vous pouvez utiliser ces commandes avec, soit des données de texte, soit des données binaires. Beaucoup de bases de données peuvent exporter des données sous forme de texte, avec les champs et les enregistrements séparés par des délimiteurs, et certaines stockent leurs données comme des fichiers texte. Les exemples de ce chapitre présentent une utilisation possible des commandes Read / Write avec des données de texte. Ces exemples supposent que vous connaissiez la structure des données stockées dans le fichier texte, comme par exemple, les délimiteurs utilisés pour séparer les champs des enregistrements. Vous pourrez utiliser ces mêmes techniques pour lire et écrire des données binaires, si vous connaissez l'organisation des données dans le fichier. Par exemple, si vous connaissez le format d'un fichier de type 'PICT', vous pourrez écrire des scripts qui liront et écriront dans des fichiers 'PICT'.

Les commandes Read et Write peuvent utiliser la marque de fichier, un marqueur utilisé par le File Manager qui indique l'octet à partir duquel les commandes Read et Write commencent à opérer. Par défaut, la marque de

fichier est positionnée sur le premier octet du fichier. Après que la commande Read ait lu une portion d'octets ou que la commande Write ait écrit sur une portion d'octets, la marque de fichier est réglée sur l'octet suivant immédiatement la fin de cette portion. Les prochaines commandes Read et Write prendront alors en compte cette nouvelle position pour débuter leurs actions.

Par exemple, supposons que vous vouliez extraire un enregistrement particulier dans une base de données de noms et d'adresses au format texte. Pour le faire, vous aurez besoin de connaître le nombre de champs de chaque enregistrement, la position de l'enregistrement désiré dans la base, et les délimiteurs utilisés pour séparer les enregistrements. Vous utiliserez alors la commande Open for Access pour obtenir le numéro de référence du fichier contenant l'enregistrement désiré, et la commande Read à l'intérieur d'une instruction Repeat pour lire successivement chaque enregistrement. Après avoir lu chaque enregistrement, la commande Read règlera la marque de fichier au début de l'enregistrement suivant. Lorsque l'instruction Repeat aura déterminé que l'enregistrement désiré a été atteint, elle retournera les données de cet enregistrement. Le script suivant montre une façon de faire tout ceci :

```
--choix du fichier à traiter
set pathToUse to choose file

try
   set x to open for access pathToUse
   set z to ReadRecord(10, 1, tab, return, x)
   close access x
   z --affichage de l'enregistrement désiré
on error errString number errNum
   display dialog errString
   close access x
end try

on ReadRecord(numberOfFields, whichRecord, fieldDelimiter, ¬
   recordDelimiter, fileRefNum)

   try
      (* s'il y a un délimiteur d'enregistrements, lire tous
         les champs excepté le dernier en utilisant le
         délimiteur de champs, puis lire le dernier champ en
         utilisant le délimiteur d'enregistrements *)
      if recordDelimiter is "" then
         set readxTimes to numberOfFields
      else
         set readxTimes to numberOfFields - 1
```

```
      end if
      repeat whichRecord times
         set recordData to {}
         repeat (readxTimes) times
            set recordData to recordData & ¬
               {(read fileRefNum before fieldDelimiter)}
         end repeat

         if readxTimes is not numberOfFields then
            set recordData to recordData & ¬
               {(read fileRefNum before recordDelimiter)}
         end if
      end repeat
      return recordData
   on error errString number errNum
      display dialog errString
      return errString
   end try
end ReadRecord
```

Ce script commence par l'utilisation de la commande Choose File afin de permettre à l'utilisateur de choisir le fichier contenant l'enregistrement désiré. Après avoir initialisé la variable dans laquelle l'enregistrement sera lu, le script utilise la commande Open for Access pour ouvrir le fichier et le gestionnaire `ReadRecord` pour lire un enregistrement spécifique.

Le gestionnaire `ReadRecord` a 5 paramètres :

`numberOfFields`	Le nombre de champs par enregistrement.
`whichRecord`	Un nombre entier qui identifie la position de l'enregistrement désiré.
`fieldDelimiter`	Le délimiteur utilisé dans le fichier pour séparer les champs.
`recordDelimiter`	Le délimiteur (s'il y en a un) utilisé pour séparer les enregistrements. Si le fichier n'utilise pas de délimiteurs différents pour séparer les enregistrements, ce paramètre doit être réglé sur "".
`fileRefNum`	Un numéro de référence de fichier obtenu avec la commande Open for Access.

Si `recordDelimiter` est réglé sur "", le gestionnaire `ReadRecord` lira le nombre spécifié de champs pour chaque enregistrement. Si `recordDelimiter` est réglé sur un délimiteur, le gestionnaire `ReadRecord` lira tous les champs

d’un enregistrement sauf le dernier, puis il lira le dernier champ jusqu’au délimiteur d’enregistrement. Cette procédure est nécessaire pour s’assurer que le dernier champ de l’enregistrement ne sera pas combiné avec le premier champ de l’enregistrement suivant.

Le gestionnaire `ReadRecord` lit chaque nouvel enregistrement dans la variable `recordData`. Si l’enregistrement est celui recherché, `ReadRecord` retournera cet enregistrement. Si l’enregistrement ne convient pas, `ReadRecord` règlera `recordData` sur une liste vide et lira l’enregistrement suivant.

Dans le script suivant, vous voulez localiser la position exacte d’un enregistrement à supprimer du fichier texte. Mais en plus de vouloir le localiser, vous voulez stocker dans une variable, tous les enregistrements le suivant, puis commencer l’écriture du contenu de cette variable au début de l’enregistrement à supprimer. Ensuite, à l’aide des commandes Get EOF et Set EOF, vous voulez obtenir la taille initiale du fichier et la réinitialiser après l’opération de suppression. Le script suivant montre une des façons de faire tout ceci :

```
-- choix du fichier à traiter
set pathToUse to choose file

try
   set x to open for access pathToUse with write permission
   DeleteRecord (10, 1, tab, return, x)
   close access x
on error errString number errNum
   display dialog errString
   close access x
end try

on DeleteRecord(numberOfFields, whichRecord, ¬
      fieldDelimiter,recordDelimiter,fileRefNum)
   try
      --initialisation des variables
      set startSize to get eof fileRefNum -- taille courante
      set idx to 1 -- compteur
      -- position de l’enregistrement à effacer
      set preRecordSize to 1
      -- taille totale des enregistrements lus
      set accumulatedSize to 0

      if recordDelimiter is "" then
         set readxTimes to numberOfFields
      else
```

```
      set readxTimes to numberOfFields - 1
   end if

   repeat with idx from 1 to whichRecord
      repeat (readxTimes) times
         set q to read fileRefNum until fieldDelimiter
         set accumulatedSize to accumulatedSize + ¬
                                  (length of q)
      end repeat

      if readxTimes is not numberOfFields then
         set q to read fileRefNum until fieldDelimiter
         set accumulatedSize to accumulatedSize + ¬
                                  (length of q)
      end if

      (* si l'enregistrement à supprimer est le premier
      enregistrement du fichier ou le prochain enregistrement
      qui sera lu, régler preRecordSize *)
      if whichRecord is 1 or idx is whichRecord - 1 then
         if whichRecord is 1 then
            set preRecordSize to 1
         else
            set preRecordSize to accumulatedSize
         end if
      end if
   end repeat

   (* maintenant que preRecordSize est déterminé, lire
      l'enregistrement devant être supprimé règle la marque
      de fichier au début de l'enregistrement suivant *)
   set fileBuffer to read fileRefNum ¬
                           from accumulatedSize + 1

   (* la suite, écrire le reste du fichier par-dessus
      l'enregistrement à supprimer *)
   if (startSize - accumulatedSize) is not 0 then
      write fileBuffer to fileRefNum ¬
                           starting at preRecordSize
      set eof fileRefNum to (startSize - accumulatedSize)
   else
      (* si le fichier contient uniquement l'enregistrement à
         supprimer, régler la fin du fichier sur 0 *)
      if whichRecord is 1 then
         set eof of fileRefNum to 0
      (* si l'enregistrement à supprimer est le dernier
      enregistrement du fichier, juste rétrécir le fichier *)
```

```
        else
           set eof of fileRefNum to preRecordSize
        end if
     end if
  on error errString number errNumber
     display dialog errString
  end try
end DeleteRecord
```

Le gestionnaire `DeleteRecord` a 5 paramètres :

`numberOfFields`	Le nombre de champs par enregistrement.
`whichRecord`	Un nombre entier qui identifie la position de l'enregistrement à supprimer.
`fieldDelimiter`	Le délimiteur utilisé dans le fichier pour séparer les champs.
`recordDelimiter`	Le délimiteur (s'il y en a un) utilisé pour séparer les enregistrements. Si le fichier n'utilise pas de délimiteurs différents pour séparer les enregistrements, ce paramètre doit être réglé sur "".
`fileRefNum`	Un numéro de référence de fichier obtenu avec la commande Open for Access.

Comme le gestionnaire `ReadRecord`, le gestionnaire `DeleteRecord` lira le nombre spécifié de champs pour chaque enregistrement si `recordDelimiter` est réglé sur "". Si `recordDelimiter` est réglé sur un délimiteur, le gestionnaire `DeleteRecord` lira tous les champs d'un enregistrement sauf le dernier, puis il lira le dernier champ jusqu'au délimiteur d'enregistrement. La taille de chaque enregistrement successif est ajoutée à la variable `accumulatedSize`, laquelle cumule la taille des enregistrements précédemment lus.

Lorsqu'il atteint l'enregistrement à supprimer, `DeleteRecord` stocke le contenu de `accumulatedSize` dans la variable `preRecordSize`, il lit entièrement l'enregistrement à supprimer et règle ainsi la marque de fichier, il lit alors de la marque de fichier jusqu'à la fin du fichier, puis il stocke cette portion de fichier dans la variable `fileBuffer`. Finalement, `DeleteRecord` écrit le contenu de `fileBuffer` en commençant l'écriture au début de l'enregistrement à supprimer.

Le script suivant montre comment utiliser des techniques similaires pour insérer un enregistrement dans une base de données texte.

```
-- choix du fichier à traiter
set pathToUse to choose file

try

(* d'abord mettre l'enregistrement à ajouter dans une variable
   ; ici, l'enregistrement à ajouter est une liste AppleScript
   car le fichier ne comporte pas d'étiquettes pour les
   données*)

   set newRecord to ¬
      {"Granny", "Smith", "123 Potato Chip Lane", ¬
      "Palo Minnow", "CA", "98761", "Snackable Computer", ¬
      "888-987-0987", "978 -234-5432", "123-985-1122"}
   set x to open for access pathToUse with write permission
   AddRecord(newRecord, 5, tab, return, x)
   close access x
on error errString number errNum
   display dialog errString
   close access x
end try

on AddRecord(recordToAdd, addWhere, fieldDelimiter, ¬
            recordDelimiter, fileRefNum)
   try
      --initialiser les variables
      set idx to 1 --compteur
      --position de l'octet à partir duquel se fait
      --l'ajout
      set preRecordSize to 1
      --taille totale des enregistrements lus
      set accumulatedSize to 0
      set numberOfFields to count of recordToAdd
      if recordDelimiter is "" then
         set readxTimes to numberOfFields
      else
         set readxTimes to numberOfFields - 1
      end if

      (* si l'enregistrement doit être ajouté au début du
         fichier, l'instruction If ajoute l'enregistrement *)
      if addWhere is 1 then
         --lire depuis le début le fichier et stockage dans
         --postBuffer
         set postBuffer to read fileRefNum from 1

         (* avant d'écrire un nouvel enregistrement, la marque
```

```
            de fichier doit être initialisée au début du fichier
            ; pour le faire, écrire une chaîne de caractères
            vide au début du fichier *)
         write "" to fileRefNum starting at 0
         WriteNewRecord(recordToAdd, fieldDelimiter, ¬
                  recordDelimiter, fileRefNum)

         --maintenant rajouter le reste de l'enregistrement
         write postBuffer to fileRefNum
         return
      end if

      (* si l'enregistrement doit être ajouté ailleurs qu'au
         début du fichier, le reste du gestionnaire AddRecord
         est exécuté *)
      repeat with idx from 1 to addWhere - 1
         repeat (readxTimes) times
            set q to read fileRefNum until fieldDelimiter
            set accumulatedSize to accumulatedSize + ¬
                   (length of q)
         end repeat
         if readxTimes is not numberOfFields then
            set q to read fileRefNum until recordDelimiter
            set accumulatedSize to accumulatedSize + ¬
                 (length of q)
         end if
      end repeat

      (* lire du début du fichier jusqu'à l'octet auquel le
         nouvel enregistrement doit être ajouté *)
      set postBuffer to read fileRefNum from ¬
         accumulatedSize + 1

      (* avant d'écrire le nouvel enregistrement, régler la
         marque de fichier sur l'octet à partir duquel le nouvel
         enregistrement doit être ajouté ; pour faire ceci,
         écrire une chaîne de caractères vide à cet octet *)
      write "" to fileRefNum starting at accumulatedSize + 1
      WriteNewRecord(recordToAdd, fieldDelimiter, ¬
           recordDelimiter, fileRefNum)

      --maintenant rajouter le reste de l'enregistrement
      write postBuffer to fileRefNum
   on error errString number errNum
      display dialog errString
   end try
end AddRecord
```

```
on WriteNewRecord(recordToAdd, fieldDelimiter, ¬
            recordDelimiter, fileRefNum)
   try
      set numberOfFields to count of recordToAdd
      if recordDelimiter is "" then
         set readxTimes to numberOfFields
      else
         set readxTimes to numberOfFields - 1
      end if
      repeat with idx from 1 to numberOfFields
         if idx £ readxTimes then
            write item idx of recordToAdd & fieldDelimiter to ¬
                  fileRefNum
         else

            (* si le fichier utilise un délimiteur
               d'enregistrement, écrire le délimiteur après le
               dernier champ dans l'enregistrement *)
            write item idx of recordToAdd & recordDelimiter to ¬
                  fileRefNum
         end if
      end repeat
   on error errString number errNum
      display dialog errString
   end try
end WriteNewRecord
```

Le gestionnaire `AddRecord` a 5 paramètres :

`recordToAdd`	Une liste de champs pour l'enregistrement à ajouter.
`whichRecord`	Un nombre entier qui identifie la position de l'enregistrement à ajouter.
`fieldDelimiter`	Le délimiteur utilisé dans le fichier pour séparer les champs.
`recordDelimiter`	Le délimiteur (s'il y en a un) utilisé pour séparer les enregistrements. Si le fichier n'utilise pas de délimiteurs différents pour séparer les enregistrements, ce paramètre doit être réglé sur "".
`fileRefNum`	Un numéro de référence de fichier obtenu avec la commande Open for Access.

Si le nouvel enregistrement doit être ajouté au début du fichier, `AddRecord` lira tous les enregistrements du fichier et les stockera dans la variable `postBuffer`, puis il initialisera la marque de fichier au début du fichier en

écrivant à cet emplacement une chaîne de caractères vide. Cette technique est très pratique pour régler la marque de fichier, lorsque vous ne souhaitez pas lire ou écrire des données.

Ensuite, `AddRecord` utilise le gestionnaire `WriteNewRecord` pour écrire l'enregistrement au début du fichier, puis il écrit le contenu de la variable `postBuffer` après cet enregistrement. Notez que la commande Write règle la marque de fichier sur la fin du fichier, donc cet exemple n'a pas besoin d'utiliser les commandes Get EOF ou Set EOF.

Si le nouvel enregistrement doit être ajouté ailleurs qu'au début du fichier, `AddRecord` utilisera une instruction Repeat pour lire tous les enregistrements précédant l'emplacement du nouvel enregistrement. Si `recordDelimiter` est réglé sur `""`, `AddRecord` lira le nombre spécifié de champs de l'enregistrement sauf le dernier, puis il lira le dernier champ jusqu'au délimiteur d'enregistrements. La taille de chaque enregistrement lu sera successivement ajoutée à la variable `accumulatedSize`, laquelle contiendra la taille totale des enregistrements précédemment lus.

Après avoir stocké, dans la variable `accumulatedSize`, la taille totale des enregistrements précédant le point à partir duquel le nouvel enregistrement doit être ajouté, `AddRecord` lira le reste du fichier et le stockera dans la variable `postBuffer`. Alors il initialisera la marque de fichier sur l'octet à partir duquel le nouvel enregistrement doit être ajouté en écrivant une chaîne de caractères vide à cet emplacement. Après avoir utilisé le gestionnaire `WriteNewRecord` pour écrire l'enregistrement, `AddRecord` écrira le contenu de la variable `postBuffer` après le nouvel enregistrement.

Le gestionnaire `WriteNewRecord` a 4 paramètres :

`recordToAdd`	Une liste de champs pour l'enregistrement à ajouter.
`fieldDelimiter`	Le délimiteur utilisé dans le fichier pour séparer les champs.
`recordDelimiter`	Le délimiteur (s'il y en a un) utilisé pour séparer les enregistrements. Si le fichier n'utilise pas de délimiteurs différents pour séparer les enregistrements, ce paramètre doit être réglé sur "".
`fileRefNum`	Un numéro de référence de fichier obtenu avec la commande Open for Access.

Si `recordDelimiter` est réglé sur "", `WriteNewRecord` ajoutera un délimiteur

de champs après chaque champ qu'il écrira. Si `recordDelimiter` est réglé sur un délimiteur spécifique, `WriteNewRecord` ajoutera un délimiteur de champs après chaque champ dans l'enregistrement sauf avec le dernier champ, il rajoutera après le dernier champ un délimiteur d'enregistrement.

Le script suivant montre une des manières de tirer avantage du fait que la commande Open for Access peut créer un fichier avec un nom spécifié à un emplacement défini si le fichier n'existe pas déjà à cet emplacement.

```
on openFileIfItExists (theFile,writePermission)
   try
      (* si le fichier n'existe pas, Info For retourne une
         erreur numéro -43 *)
      set x to info for file theFile
      if writePermission is true then
         return (open for access file theFile ¬
              with write permission)
      else
         return (open for access file theFile)
      end if
   on error theErrMsg number errorNum
      try
         (* si erreur -43, l'utilisateur peut choisir de créer
            le fichier *)
         display dialog "Le fichier : " & theFile & ¬
         "n'existe pas." buttons {"Création","Annuler","Ok"} ¬
         default button 2
         if button returned of the result is "Ok" then
            return errorNum
         else
            --création du fichier
            if writePermission is true then
               return open for access file theFile ¬
                    with write permission
            else
               return open for access file theFile
            end if
         end if
      on error theErrMsg number theErrNumber
         return theErrNumber
      end try
   end try
end openFileIfItExists

(* régler une variable sur le fichier que vous voulez ouvrir ou
   créer *)
set fileToOpenOrCreate to "Disque Dur:Test File One"
```

```
set z to openFileIfItExists(fileToOpenOrCreate,true)
if z < 0 then
   --openFileIfItExists a retourné une erreur
   display dialog the result
else
   (* openFileIfItExists a retourné un numéro de référence de
      fichier, vous pouvez maintenant travailler avec à partir
      d'ici *)
   close access z
end if
```

Le gestionnaire `openFileIfItExists` a 2 paramètres :

`theFile`	Une chaîne de caractères qui correspond au chemin entier du fichier à ouvrir ou à créer.
`writePermission`	Une valeur booléenne qui indique s'il faut ouvrir le fichier avec (`true`) ou sans (`false`) autorisation d'écriture.

Pour déterminer si le fichier existe ou pas, `openFileIfItExists` utilise la commande Info For. Si le fichier n'existe pas, la commande Info For retournera une erreur -43 (Le fichier <name> est introuvable.), et `openFileIfItExists` affichera une boîte de dialogue qui permettra à l'utilisateur de choisir s'il faut créer ou non le nouveau fichier. Si le fichier existe ou s'il est créé, `openFileIfItExists` l'ouvrira avec ou sans autorisation d'écriture, en fonction de la valeur du paramètre `writePermission`.