FileXtra3

A free add-on for Director Lingo Programmers

Table of Contents

Contact Info

Contact the author at kent@kblab.net.

The most current versions of my Xtras will be at http://www.kblab.net/xtras.

FileXtra3 is provided on an "as-is" basis, which basically means I assume no responsibility for how it is used and have no liability if it does not suit your needs.

Target Platforms

FileXtra3 is designed to run on Macromedia Director versions 6 and newer for both the Power Macintosh using System 8.5 or newer and Windows using Windows 95 OSR2, 98, 98 SE, Me; NT 4 or Windows 2000. Original Windows 95 is not supported. You must have OSR2 if you are using Win 95.

Windows 95 and NT users need to have at least Internet Explorer 4.0 installed. This guarantees current enough versions of critical system DLL's are present for certain methods to work properly.

Note that FileXtra3 is very definitely NOT Shockwave-safe! Its ability to write information to permanent storage via some of its method calls means that it cannot be considered safe and is not packaged that way.

How To Report Bugs

Send an email to the author at kent@kblab.net. PLEASE provide the following information:

- the platform (Mac, PC, both)
- Director version
- OS version
- how much RAM you've got
- any error codes returned by FileXtra3 methods.

Whither FileXtra?

FileXtra3 is a no-cost, cross-platform Lingo scripting Xtra for Macromedia Director versions 6 and newer. You are free to use it in your projects and products at absolutely no cost. The only stipulation I would place on you is that if you share the Xtra with someone who is a developer, please give them the documentation as well.

The first version of the Xtra was called FileUtil and was released in 1996, soon after the release of Director v5. I wrote the original version because I had migrated a commercial children's educational application from Director 4 to v5 to take advantage of the spiffy new features and Windows 95's better memory management.

However upon performing this migration, I realized that some of the "XObjects" that worked with Director 4 did not work with v5. This in effect crippled the application I was coding by taking away certain functionality that only the XObjects provided.

So I madly scrambled to try and learn how to create these new extensions called "Xtras." Macromedia provided a developer's kit along with some example code. A few weeks later I produced FileUtil. At the time there were very few Xtras available. A west coast company contacted me and asked if I would be interested in letting them resell the Xtra, which I agreed to on a non-exclusive basis.

After a few months I decided to pull the plug on this arrangement as it was inconvenient and costly to users. I was more interested in granting free access to my Xtras so that they could be used by anyone that needed the functionality.

Another reason for giving the Xtra away is that when I started Director programming with version 4, there were many free XObjects to help developers like myself produce useful products. It became more important to give something back to the community that helped me than to make a few dollars.

So I added some functionality and changed the name to FileXtra, and made it freely available to anyone that needed it. According to the email I have received, it has been helpful to many other developers in many other countries as well as here in the U.S. So I hope that if you are a new user you will find it useful also.

Differences From Previous Versions

If you have used an earlier version of FileXtra, you will find all of the old methods in the new release.

However, many things have fundamentally changed with the new release, including:

- you must now instantiate the Xtra before use, i.e. there are no longer any "global" methods
- the return values from methods are now True/False instead of error codes, with the exception of methods that need to return a character string or a list
- error reporting methods are now provided
- all of the method names now begin with "fx_" so they will be immediately recognizable in your code, and you won't get FileXtra3 methods confused with other Xtras such as FileIO

I decided to require the Xtra be instantiated because in certain low-memory situations it is helpful to be able to free up unneeded memory. This is also "cleaner" from an object-oriented perspective as objects that are no longer in use can be destroyed and their memory reclaimed.

The previous versions of the Xtra returned any number of error codes when executed as a negative number. This was great for an old C programmer like myself, but was unnecessarily bulky and confusing for others. So FileXtra3 simply returns True (1) if a method succeeded and False (0) if it failed. The exceptions to this are noted in the documentation and are mostly confined to methods that must return a string (such as fx_FileOpenDialog) or a list (such as fx_VolumesToList).

Since I now return True/False from methods, some mechanism was needed to report on errors that occurred. fx_ErrorNumber will report the error code from the last method that was executed, while fx_ErrorString will return a human-readable interpretation of what happened.

Finally I prefixed all methods with "fx_" to make it perfectly clear that this was a call to FileXtra3. There are enough Xtras out there now as to lead to confusion about what is being called in some cases. Also the method names have been changed to make it clear what they are doing and what "group" they belong to. So all of the file methods begin with "fx_File", all of the folder methods begin with "fx_Folder", all of the link methods begin with "fx_Link", all of the volume methods begin with "fx_Volume" and all of the error methods begin with "fx_Error."

Note also that <u>it is possible to use the previous version of the Xtra at the same time as FileXtra3</u> because of these naming differences. This was done to aid in migration of projects to the new release.

Bug Fixes

Unfortunately, as with any software project, there are problems that can occur with the code. Below is a list of the known bugs and their fixes.

Bugs fixed in 3.0

- files copied from CD-ROM tagged as read only
- drive sizes over 2 GB reported incorrectly
- file open & save dialogs not modal
- file open & save dialogs not starting in desired directory (Macintosh only)
- fx_FolderExists now correctly reports root directories of volumes as folders (Macintosh only)
- UNC volume names on Windows caused problems
- maximum path length has been increased on the Macintosh from 256 to 1024 characters.

Previously FileXtra when copying files to a new location would replicate the file's attributes, including the read-only flag. Now in all cases when doing file copies FileXtra3 will make the destination files read-write. This particular problem caused a great deal of angst among those using Director as a tool for writing installers, with good reason.

Since FileXtra was first written, hard drive sizes have increased dramatically. New OS calls exist to correctly report the sizes of these drives and FileXtra3 takes advantage of them. Note however that Director's "integer" type is not sufficiently large to handle these numbers, so double-precision floating point numbers are returned instead.

The standard OS file open and save as dialog boxes are now modal. This solves another vexing problem in that with previous versions of the Xtra, it was possible for a user to click outside of the dialog box and make it hidden behind the Director stage, with no clear way to get it back.

On the Macintosh I have now figured out how to make the file open and save as dialog boxes start in the desired directory. My thanks to Apple Computer for providing sample code as to the incredibly convoluted way to do this.

With previous versions of the Xtra on the Macintosh using DirectoryExists("some volume name") would report an error, whereas if you had appended a ":" to the end of the volume name it would have worked. fx_FolderExists will correctly report the existence of the root directory (folder) in either case.

UNC named volumes on Windows are now supported by FileXtra3. (UNC stands for Universal Naming Convention).

Bugs fixed in 3.01

Below is a list of the known bugs fixed in 3.01. My thanks to everyone who reported problems and provided the necessary details to zap these gremlins.

method	platform(s)	problem
*	Windows NT 4	The Xtra could not load because it was looking in shell32.dll (a system DLL) for a function that only existed in Windows 2000, Windows Me, Windows 98 or systems with the Internet Explorer 4 "service pack 1" Desktop Update package. (Fixed by using a different combination of system calls.)
fx_FolderSelectDialog	Macintosh	Crashes machine when a root volume is chosen. (Fixed)
fx_FolderSelectDialog	Windows	When a root volume is chosen (such as C:), it appended an extraneous '\' character. (Fixed)
fx_FileSaveAsDialog	Macintosh	If you did not append a ':' to the end of your 'initialPath' argument it would not start in the correct directory. (Fixed)
fx_FileCopy	all	FileXtra3 would allow you to try and copy a file onto itself, with bad results such as the truncation of the file to 0 bytes. (Fixed)

New Features

There are many new features in this release of FileXtra, including a folder selection dialog, support for alias/shortcut files, obtaining “special paths” from the system, recycling files and folders, moving files and folders, opening and printing documents, synchronizing folder contents and ejecting removable media, among others.

A listing of new methods is provided below:

- fx_GetVersion – returns the version of the Xtra
- fx_FileIsLink – tells whether the given file is really a “link” (alias/shortcut)
- fx_FileRecycle – move file(s) to the trash/recycle bin instead of immediate deletion
- fx_FileMove – relocate a file without copying & deleting
- fx_FileGetWriteState – get read-only flag for a file
- fx_FileSetWriteState – set or clear the read-only flag for a file
- fx_FileGetSize – returns the number of bytes a file uses on disk
- fx_FileGetType – get a file’s “type”
- fx_FileSetType – set a file’s “type”
- fx_FileCompare – compare two versions of a file to see if they are the same
- fx_FileOpenDocument – open a document
- fx_FilePrintDocument – print a document
- fx_FileGetAppPath – get the path to the application for the specified file type
- fx_FileRunApp – run an application
- fx_LinkCreate – create an alias (Mac) or shortcut (Windows) to a file
- fx_LinkResolve – find out what a link points to
- fx_FolderSelectDialog – pick a folder from a dialog
- fx_FolderGetSpecialPath – find the path to special system folders like the Desktop
- fx_FolderRecycle – send a folder and its contents to the Trash or Recycle Bin
- fx_FolderMove – relocate a folder and its contents without copying & deleting
- fx_FolderSyncOneWay – synchronize a folder’s contents in one direction only
- fx_FolderSyncBothWays – synchronize two folder’s contents so they match
- fx_VolumeGetTotalBytes – total number of bytes on a drive
- fx_VolumeIsRemovable – tests whether a volume uses removable media
- fx_VolumeEject – programmatically unmount & eject volumes such as CD-ROMs
- fx_ErrorNumber – error code returned by most recent FileXtra3 method call
- fx_ErrorString – human readable message about most recent error

Cross-Platform Information

I have always produced cross-platform Director products, usually burned on hybrid CD-ROMs with both Macintosh and PC versions. So naturally all of the code that I write, especially Xtras, must work on both platforms. Not only that, but they must function as identically as possible between the two platforms so that I don't have to write a lot of "special-case" Director code to work around the gaps or limitations.

If it sounds like I'm lazy, I'm not, I just don't like having to write a lot of extra code. Code that needs extra development time. Code that needs extra testing.

FileXtra3 does an excellent job of providing identical functionality across platforms. Every method is implemented on both platforms, not just Mac or Windows. There's nothing I hate worse when using somebody else's Xtra for a project than to discover that a neat feature I want to use only works on one platform!

Path Specification

FileXtra3 depends completely upon volume, folder and file "paths." These paths are specified the same between platforms with the exception of the path "separator" character. This character is a colon (':') for the Macintosh and a backslash ('\') for Windows.

Windows volumes typically start with a drive letter, such as 'A' or 'C', and include a colon. To specify the root or top-most directory on a Windows machine you would use something like "C:\".

To refer to shared or network volumes on Windows you use something called UNC names. While these UNC-named volumes can be "mapped" to drive letters on a Windows machine, FileXtra3 does not require that. FileXtra3 works with UNC-named volumes, such as "\\LinuxBox\kkersten\details.doc".

Macintoshes of course do not have this split personality when it comes to naming volumes, whether they be local or networked. A typical Macintosh pathname might be "Linux Server:Kent's Director files:details.doc".

FileXtra3 does not care if you do or do not append '\' or ':' characters to the end of paths that end with a folder name. The exception to this rule is for Windows, you must specify the root directory as C:\ instead of C:.

How To Use The Xtra

To use FileXtra3, place it into your “Xtras” folder that resides in the same folder as your Director executable.

To see a list of the methods available in the Message window, type the following:

```
put xtra(“FileXtra3”).interface()
```

To “instantiate” or create an instance of the xtra, use the following code:

```
fxObj = xtra(“FileXtra3”).new()
```

After instantiation, you can use any of the xtra’s methods as many times as you like.

I typically use the new “dot” syntax (available since at least Director 7) in my method calls, like so:

```
fooStr = fxObj.fx_GetVersion()
```

Although you could also write the previous line as:

```
fooStr = fx_GetVersion(fxObj)
```

When you are done using the xtra, free up its memory with the following code:

```
fxObj = 0
```

Informational Methods

There is currently only one informational method, and that returns the current version of the xtra.

fx_GetVersion

fx_GetVersion

Name

fx_GetVersion – return the version of the xtra

Synopsis

strVar = fx_GetVersion(object me)

Description

This method returns a string that represents the version of FileXtra3 that is in use.

Return Type

String

Macintosh Notes

None.

Windows Notes

None.

Example

Using “dot” syntax:

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_GetVersion()
-- "3.01 FileXtra of 9-Nov-2000 (c) 1996-2000 by Kent Kersten"
fxObj = 0
```

Using the old (but still acceptable) syntax:

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_GetVersion()
-- "3.01 FileXtra of 9-Nov-2000 (c) 1996-2000 by Kent Kersten"
fxObj = 0
```

Error Codes

None.

File Methods

File methods operate on a file or group of files. There are methods available to check for the existence of a file, copy files, rename files, delete files, move files to the Trash or Recycle Bin, and others.

- fx_FileOpenDialog
- fx_FileSaveAsDialog
- fx_FileExists
- fx_FileIsLink
- fx_FileRename
- fx_FileDelete
- fx_FileRecycle
- fx_FileCopy
- fx_FileMove
- fx_FileGetWriteState
- fx_FileSetWriteState
- fx_FileGetModDate
- fx_FileGetSize
- fx_FileGetType
- fx_FileSetType
- fx_FileCompare
- fx_FileOpenDocument
- fx_FilePrintDocument
- fx_FileGetAppPath
- fx_FileRunApp

fx_FileOpenDialog

Name

fx_FileOpenDialog – return the chosen filename from the File Open dialog

Synopsis

Macintosh: strVar = fx_FileOpenDialog(object me, string initialFolder, string filtStr)

Windows: strVar = fx_FileOpenDialog(object me, string initialFolder, string filtStr, string dlogTitle, Boolean createPrompt, Boolean fileMustExist)

Description

This method returns a filename that the user chooses from a standard system File Open dialog box. If no file is chosen, such as when the Cancel button is pressed, the empty string (“ ”) is returned instead.

If a valid file is chosen, the complete path to the file is returned.

initialFolder is the path of the folder where the dialog should point to when opened.

filtStr is a string that tells what kind of files to show in the dialog. On Windows, these consist of descriptor/extension pairs separated by `/', such as “Text Files/*.TXT/All Files/*.*”. On the Macintosh, the filters are file `types' separated by `/', such as “TEXT/WORD”. There is no limit to the number of filters on Windows, but you can specify a maximum of four (4) filters on the Macintosh.

dlogTitle (Windows only) is the title of the Windows dialog. If you pass “ ”, the title defaults to “Open.”

createPrompt (Windows only) is a boolean that tells the Windows dialog to prompt the user about creating the file if the file does not already exist if you specify True. For instance, the user can type the name of a file into the text box; if that file does not exist when they press the “Open” button, a dialog will appear asking them if they wish to create it (only if you specified True). If they answer “No”, the Open File dialog stays on the screen. If they answer “Yes”, the filename is returned to the caller. Remember that you can use the fx_FileExists() method to see if the file actually exists or not. Pass False to disable this feature.

fileMustExist (Windows only) is a boolean that tells the Windows dialog that a user must either select a file from the list or type the name of an existing file if this argument is True. If they do not, a message will appear asking them to try again. Pass False to disable this feature.

If an error occurs, the empty string “ ” is returned.

Return Type

String

Macintosh Notes

None.

Windows Notes

None.

Example

Macintosh:

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileOpenDialog("Macintosh HD:" "TEXT/WORD")
-- "Macintosh HD:Documents:Word docs:File Formats"
fxObj = 0
```

Windows:

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileOpenDialog("C:\My Documents\" "Text Files/*.TXT",
            "Choose a text file", True, True)
-- "C:\My Documents\MISC\Nowxport.txt"
fxObj = 0
```

Error Codes

None.

fx_FileSaveAsDialog

Name

fx_FileSaveAsDialog – return a filename entered in a Save As dialog

Synopsis

Macintosh: strVar = fx_FileSaveAsDialog(object me, string initialFolder, string filename, string prompt)

Windows: strVar = fx_FileSaveAsDialog(object me, string initialFolder, string filename, string dlogTitle, Boolean overwritePrompt)

Description

This method displays a system File Save As dialog box that allows the user to select a directory and type in a filename to save a file under. The full path including the filename are returned to the caller if the Save button was pressed. The empty string (“ ”) is returned if Cancel was pressed.

initialDir is the path of the directory where the dialog should be opened.

filename is the name to show initially in the dialog box. The user can change this by typing over it.

prompt (Macintosh only) is the text that appears above the name of the file. If you leave this blank, it defaults automatically to “Save As:”.

dlogTitle (Windows only) will be used as the title bar text of the Windows dialog.

overwritePrompt (Windows only) is a boolean that, when True, brings up a warning dialog if the user types in the name of an existing file and presses the Save button. If this flag is False then no warning is given.

If an error occurs, the empty string “ ” is returned.

Return Type

String

Macintosh Notes

None.

Windows Notes

None.

Example

Macintosh:

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_FileSaveAsDialog(“Macintosh HD:”, “myfile”, “Enter a filename:”)
-- “Macintosh HD:myfile”
fxObj = 0
```

Windows:

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_FileSaveAsDialog("c:\temp", "myfile.jpg", "Enter a filename", True)
-- "C:\Temp\myfile.jgp"
fxObj = 0
```

Error Codes

None.

fx_FileExists

Name

fx_FileExists – check for the existence of a file

Synopsis

intVar = fx_FileExists(object me, string fileName)

Description

This method checks to see if ***fileName*** exists. You should of course as with all FileXtra3 methods use a complete path specification consisting of volume name, folder name(s) and the file name.

Returns True (1) if the file exists, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

You can pass wildcards for the filename. If any files match True (1) will be returned. If no files match, False (0) is returned.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileExists("Macintosh HD:pillow fight.doc")
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win

fx_FileIsLink

Name

fx_FileIsLink – tell whether the given file is actually a link

Synopsis

intVar = fx_FileIsLink(object me, string fileName)

Description

To determine if ***fileName*** is actually a link (alias/shortcut), use this method.

Remember that a link is still a file, even if it is a link to a folder.

Returns True (1) if the file is a link, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

On Windows, the only thing that determines if a file is possibly a link is that its name ends with ".lnk". This extension is hidden from the user in Windows Explorer, but you can see it if you do a fx_FolderToList() call.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileIsLink("c:\temp\Shortcut to noodles.doc.lnk")
-- 1
put fxObj.fx_FileIsLink("c:\temp\Shortcut to noodles.doc")
-- 0
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win
-150	Specified link file is actually a normal file	both

fx_FileRename

Name

fx_FileRename – rename a file

Synopsis

intVar = fx_FileRename(object me, string oldName, string newName)

Description

Renames ***oldName*** to ***newName***.

Returns True (1) if successful, False (0) if not or if an error occurs.

Attempting to rename a file to an existing folder name is considered naughty.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_FileRename(“Macintosh HD:frooby”, “Macintosh HD:towel”)
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-1	General error of unknown origin	both
-6	File rename failure	Mac
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win
-210	New filename already exists or two paths are different	Win

fx_FileDelete

Name

fx_FileDelete – delete file(s)

Synopsis

intVar = fx_FileDelete(object me, string fileName)

Description

Delete a single file (Macintosh) or a group of files (Windows – with wildcards).

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

You can use wildcards. Wildcards on Windows systems are '*' for match 0 or more characters and '?' for match any one character. So to catch all files you would use "*.*". If you wanted to narrow it more you could use something like "picture?.jpg" to catch files named picture0.jpg – picture9.jpg.

Example

Macintosh:

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileDelete("Boot:plugh")
-- 1
fxObj = 0
```

Windows:

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileDelete("D:\*.tmp")
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-5	File deletion failure	both
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-18	Could not delete specified folder	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win

fx_FileRecycle

Name

fx_FileRecycle – place a file in the Trash/Recycle Bin

Synopsis

intVar = fx_FileRecycle(object me, string fileName)

Description

If for some reason you do not wish to immediately delete a file you can use this method to place it into the systems Trash (Macintosh) or Recycle Bin (Windows).

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

This call will handle wildcards but will not operate recursively.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileRecycle("C:\Gates\world domination plans.doc")
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win
-93	FileRecycle failed	Win
-140	Special folder type specified is unknown	Mac
-141	FindFolder() system call failed	Mac

fx_FileCopy

Name

fx_FileCopy – copy file(s)

Synopsis

intVar = fx_FileCopy(object me, string fromFName, string toFName)

Description

Macintosh: Copy a single file to a new name and/or location.
Windows: Copy single or multiple files using wildcards.

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

You can use wildcards.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileCopy("Macintosh HD:Annapurna.doc", "Toad Hall:expedition plan.doc")
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-1	General error of unknown origin	both
-5	File deletion failure	both
-7	File not found	both
-8	Specified file is actually a folder	both
-9	File creation failure	both
-10	File open failure	Mac
-11	File write failure	Mac
-13	File read failure	Mac
-16	Specified folder is actually a file	both
-17	Folder creation failure	both
-26	Cannot copy a file onto itself	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win

fx_FileMove

Name

fx_FileMove – move a file to a new location

Synopsis

intVar = fx_FileMove(object me, string fromFName, string toFName)

Description

Instead of copying a file to a new location and deleting the old file, you can use this method to move the original file instead. This would also be much faster than copy/delete.

fromFName refers to the source file to move.

toFName refers to the target file pathname.

You can only move a file within its current volume (drive). You cannot move a file to another volume with this method.

You cannot move a file if a file already exists at the destination with the given name.

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_FileMove(“c:\urgent.ppt”, “c:\BitBucket\urgent.ppt”)
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-20	I/O error	Mac
-21	Hardware volume lock	Mac
-22	Software volume lock	Mac
-23	Target directory is locked	Mac
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win
-91	Destination file already exists	both
-92	FileMove failed	Win
-95	Attempt to move into offspring	Mac

fx_FileGetWriteState

Name

fx_FileGetWriteState – tell if the file is read-only

Synopsis

intVar = fx_FileGetWriteState(object me, string fileName)

Description

Return True (1) if ***fileName*** is writeable and False (0) if it is read-only or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_FileGetWriteState(“c:\config.sys”)
-- 0
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win

fx_FileSetWriteState

Name

fx_FileSetWriteState – return the version of the xtra

Synopsis

intVar = fx_FileSetWriteState(object me, string fileName, Boolean writeable)

Description

You can set ***fileName*** to be writeable by passing True (1) for the ***writeable*** parameter. To make the file read-only, pass False (0) for writeable.

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileSetWriteState("c:\config.sys", False)
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win

fx_FileGetModDate

Name

fx_FileGetModDate – return the last modified date for a file

Synopsis

stringVar = fx_FileGetModDate(object me, string fileName)

Description

This method returns the last modified time & date for ***fileName*** as a 25-character string as follows: "Wed Jan 02 02:03:55 1980\n". The `\n' is a newline character (ASCII 13). Note that the time is in 24-hour format and numbers are zero-padded. I have done it this way because that is the format that UNIX system calls return, and being a UNIX/Linux head, it naturally seemed the best way.

If an error occurs, the null string "" is returned.

Return Type

String

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileGetModDate("Macintosh HD:Testuser")
-- "Thu Oct 26 14:33:01 2000
"
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-1	General error of unknown origin	both
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win

fx_FileGetSize

Name

fx_FileGetSize – return the size in bytes of a file

Synopsis

floatVar = fx_FileGetSize(object me, string fileName)

Description

Return the size, in bytes, of ***fileName***. If an error occurs, 0.0000 is returned.

The xtra must return the size as a floating-point number because Director integers are limited to 2^{31} in size, which is roughly 2 GB.

Return Type

Float

Macintosh Notes

The size of a Macintosh file is limited to 2^{31} bits, or around 2 GB.

Windows Notes

Works for files > 2 GB in size.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileGetSize("C:\Program Files\Macromedia\Director 8\director.exe")
-- 4775936.0000
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-1	General error of unknown origin	both
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win

fx_FileGetType

Name

fx_FileGetType – return the type of a file

Synopsis

stringVar = fx_FileGetType(object me, string fileName)

Description

A file’s type determines what application can open or print that file.

On Macintosh, the value returned is an 8-byte character string that is the type and creator put together, as in “TTTTCCCC”.

On Windows, the value returned is the file’s extension with the leading period intact, as in “.txt” or “.html”.

The empty string “” is returned if an error occurs.

Return Type

String

Macintosh Notes

None.

Windows Notes

None.

Example

Macintosh:

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_FileGetType(“Macintosh HD:Documents:project notes”)
-- “W8BNMSWD”
fxObj = 0
```

Windows:

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_FileGetType(“C:\kent.html”)
-- “.html”
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-71	No file type found for specified file	Win

fx_FileSetType

Name

fx_FileSetType – set the type of a file

Synopsis

intVar = fx_FileSetType(object me, string fileName, string fileType)

Description

Call this method to change the ***fileType*** for ***fileName***. Be careful doing this as you can disassociate a file from the application that created it or can open/print it.

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

The type is actually the "type" and "creator" put together in a single 8-byte character array.

Windows Notes

File types are the extension with a leading period, such as ".JPG" or ".TXT".

Example

Macintosh:

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileCopy("NT Server:product plan.doc", "Macintosh HD:Product Plan")
-- 1
put fxObj.fx_FileSetType("Macintosh HD:Product Plan", "W8BNMSWD")
-- 1
fxObj = 0
```

Windows:

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileSetType("C:\testfile.txt", ".doc")
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-1	General error of unknown origin	both
-6	File rename failure	Mac
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win
-71	No file type found for specified file	Win
-210	New filename already exists or two paths are different	Win

fx_FileCompare

Name

fx_FileCompare – compare two versions of a file

Synopsis

intVar = fx_FileCompare(object me, string fileName, string fileName2)

Description

This method will compare two versions of a file. If they match, True (1) is returned and False (0) if not, or if an error occurs. The files match if they have the same file size in bytes and the same modification date.

If the return value is False then check fx_ErrorNumber to see why they did not match or to get an error code.

Return Type

String

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_FileCompare(“Boot:trip plans”, “offline:trip plans”)
-- 0
put fxObj.fx_ErrorNumber()
-- -103
put fxObj.fx_ErrorString()
-- "file two's mod date is newer than file one's"
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-1	General error of unknown origin	both
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win
-101	File sizes are different	both
-103	File two’s mod date is newer than file one’s	both
-105	File one’s mod date is newer than file two’s	both

fx_FileOpenDocument

Name

fx_FileOpenDocument – open the given document

Synopsis

intVar = fx_FileOpenDocument(object me, string fileName)

Description

Given a path to a file, this method tries to determine the application that the file was created with or is associated with. If found, the application is invoked and the document opened.

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileOpenDocument("c:\Xeno2000.ppt")
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-30	SHGetSpecialFolderLocation() call failed (bummer)	Win
-42	Could not obtain Finder information for file	Mac
-44	Not enough memory to launch application	Mac
-51	Specified volume does not exist	both
-71	No file type found for specified file	Win
-73	No application associated with specified file type	Win
-74	No \\shell\\open\\command key found for specified file type	Win
-75	No \\shell\\print\\command key found for specified file type	Win
-77	Problems reading desktop database	Mac
-81	Specified application was not found; process not created	Win
-122	Could not create FSSpec record	Mac

fx_FilePrintDocument

Name

fx_FilePrintDocument – print the given document

Synopsis

intVar = fx_FilePrintDocument(object me, string fileName)

Description

Given a path to ***fileName***, this method tries to determine the application that the file was created with or is associated with. If found, the application is invoked and the document printed.

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FilePrintDocument("c:\Xeno2000.ppt")
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-30	SHGetSpecialFolderLocation() call failed (bummer)	Win
-42	Could not obtain Finder information for file	Mac
-44	Not enough memory to launch application	Mac
-51	Specified volume does not exist	both
-71	No file type found for specified file	Win
-73	No application associated with specified file type	Win
-74	No \\shell\\open\\command key found for specified file type	Win
-75	No \\shell\\print\\command key found for specified file type	Win
-77	Problems reading desktop database	Mac
-81	Specified application was not found; process not created	Win
-122	Could not create FSSpec record	Mac

fx_FileGetAppPath

Name

fx_FileGetAppPath – return the path to the application associated with a given file type

Synopsis

stringVar = fx_FileGetAppPath(object me, string fileType)

Description

Given a ***fileType***, return a string that represents the path to the application that is associated with that type.

Returns the empty string “” if an error occurs or if no application is associated with the given type.

Return Type

String

Macintosh Notes

The fileType argument is an 8-byte character array of type & creator, as in “TTTTCCCC”. We are really only using the creator bytes since we already know we are looking for a file with a type of “APPL”.

Desktop databases are scanned for the answer to the question of what application runs the particular documents using the given creator bytes. These databases are scanned in the order of when volumes were mounted, so if SimpleText for example is on two drives, it will find the one on the volume that was mounted first.

Please note that the case of the letters for type & creator matter! A creator of “MD01” will successfully find Director 8 while “md01” will not.

Windows Notes

The registry on Windows NT/2000 contains entries containing symbolic variables %SystemRoot% and %windir% for certain applications, usually Microsoft’s. FileXtra3 attempts to filter these and resolve them to the correct paths.

Note that the registry entries for some applications are surrounded by double quotes, usually if they contain spaces in the path name. This is unfortunately not true in every case.

Example

Macintosh:

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_FileGetAppPath(“APPLMD00”)
-- “Macintosh HD:Applications:Director 7:Director 7.0”
fxObj = 0
```

Windows:

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_FileGetAppPath(“.dir”)
-- ""C:\Program Files\Macromedia\Director 8\Director.exe""
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-30	SHGetSpecialFolderLocation() call failed (bummer)	Win
-73	No application associated with specified file type	Win
-74	No \\shell\\open\\command key found for specified file type	Win
-75	No \\shell\\print\\command key found for specified file type	Win
-77	Problems reading desktop database	Mac

fx_FileRunApp

Name

fx_FileRunApp – run an application

Synopsis

intVar = fx_FileRunApp(object me, string commandLine)

Description

Pass this method an application name and it will attempt to have the OS launch it.

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

You can only pass the path of an application to run. No arguments are allowed.

Windows Notes

You can pass an entire command line including the path to the application and any arguments, such as the paths of files to open.

Example

Macintosh:

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_FileRunApp(“Macintosh HD:Applications:Director 7:Director 7.0”)
-- 1
fxObj = 0
```

Windows:

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_FileRunApp(“C:\Program Files\Macromedia\Director 8\director.exe
                        c:\test.dir”)
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-44	Not enough memory to launch application	Mac
-51	Specified volume does not exist	both
-81	Specified application was not found; process not created	Win
-122	Could not create FSSpec record	Mac

Alias/Shortcut Methods

Files that refer to other files or folders but are not copies of those originals are called aliases on the Macintosh and shortcuts on Windows. FileXtra3 allows you to make these “link” files and also to determine the original file that they refer to.

fx_LinkCreate
fx_LinkResolve

fx_LinkCreate

Name

fx_LinkCreate – create a link to a file or folder

Synopsis

intVar = fx_LinkCreate(object me, string fileName, string destFolder)

Description

A “link” is an alias (Macintosh) or shortcut (Windows) file that points to another file or folder. This concept is useful if you need or want only one copy of a file or folder but need it represented in more than one place.

fileName refers to a file or folder path that the link will point to.

destFolder refers to a folder name, <u>not</u> a filename. This folder is where the link file will be created. Giving a complete pathname including filename will result in an error –126.

Please note that a link file is merely a “pointer” to the real file or folder, it does not contain the contents of the original. So for example deleting the link file will not delete the original file or folder that it points to.

Also keep in mind that a link to a folder is still a file. The respective Operating Systems give the illusion to the user that a link to a folder “looks like” a folder, but it is actually a file. Check this by using a fx_FileIsLink() method call.

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

The link file’s name has “ alias” appended. If you need to rename the alias, there are no OS reasons why you have to keep this word in the filename.

Windows Notes

The link file’s name is the name of the file with “Shortcut to “ prepended. Use fx_FileRename if you need to name it something else, but remember that the filename MUST end with “.lnk” or Windows won’t know that it is a shortcut.

Also note that when you see a shortcut on a Windows system, Windows Explorer will not show the “.lnk” extension. But if you do a fx_FolderToList() you will see the filename with the .lnk appended.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_LinkCreate("c:\rfc959.txt", "c:\temp")
-- 1
put fxObj.fx_FileExists("c:\temp\Shortcut to rfc959.txt")
-- 0
put fxObj.fx_FileExists("c:\temp\Shortcut to rfc959.txt.lnk")
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-5	File deletion failure	both
-7	File not found	both
-8	Specified file is actually a folder	both
-15	Folder not found	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win
-122	Could not create FSSpec record	Mac
-123	Could not create FSSpec record	Mac
-124	NewAlias() toolbox call failed	Mac
-125	NewAlias() toolbox call returned nil	Mac
-126	Creating resource fork of alias file failed	Mac
-127	Opening resource fork of alias file failed	Mac
-128	AddResource() on alias file failed	Mac
-129	WriteResource() on alias file failed	Mac
-130	CloseResFile() on alias file failed	Mac
-161	SetPath system call failed	Win
-162	SetDescription system call failed	Win
-163	IPersistFile::Save system call failed	Win

fx_LinkResolve

Name

fx_LinkResolve – determine what a link file refers to

Synopsis

intVar = fx_LinkResolve(object me, string fileName)

Description

Use this method to determine the target of a link file.

fileName is the name of the link file.

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_LinkResolve("c:\temp\Shortcut to rfc959.txt.lnk")
-- "C:\RFC959.TXT"
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-7	File not found	both
-8	Specified file is actually a folder	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win
-122	Could not create FSSpec record	Mac
-127	Opening resource fork of alias file failed	Mac
-130	CloseResFile() on alias file failed	Mac
-152	Could not read the alias resource	Mac
-154	ResolveAlias() failed	Mac
-155	Could not resolve alias path	Win

Folder Methods

This group of methods operates on folders (directories). Methods are available to create, delete, copy recycle, move and synchronize folders and their files.

fx_FolderSelectDialog
fx_FolderGetSpecialPath
fx_FolderExists
fx_FolderCreate
fx_FolderDelete
fx_FolderRecycle
fx_FolderCopy
fx_FolderMove
fx_FolderSyncOneWay
fx_FolderSyncBothWays
fx_FolderToList

fx_FolderSelectDialog

Name

fx_FolderSelectDialog – select a folder from a dialog box

Synopsis

Macintosh: stringVar = fx_FolderSelectDialog(object me, string initialFolder)
Windows: stringVar = fx_FolderSelectDialog(object me, string infoString)

Description

If you need to have the user select a folder and not a file from a modal dialog, use this method.

initialFolder (Macintosh only) gives a path to start the dialog in.

infoString (Windows only) allows you to specify an informational string in the dialog.

Returns a path to the folder selected, or the empty string “” if Cancel is chosen.

Returns the empty string “” if an error occurs.

Return Type

String

Macintosh Notes

You can specify the path to start the dialog in.

The returned pathname has a ‘:’ appended.

Windows Notes

You can specify an informational string to display, although it is of limited usefulness.

Windows by default starts the select dialog at the root of your system, “My Computer.”

The returned pathname has a ‘\’ appended.

Example

Macintosh:

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FolderSelectDialog("Europa:Xtra")
-- "Europa:Xtra:"
fxObj = 0
```

Windows:

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FolderSelectDialog("Choose a folder:")
-- "C:\My Documents\Music\"
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-56	SHBrowseForFolder failed	Win
-57	SHGetPathFromIDList failed	Win

fx_FolderGetSpecialPath

Name

fx_FolderGetSpecialPath – find the paths to all kinds of useful system folders

Synopsis

stringVar = fx_FolderGetSpecialPath(object me, string folderType)

Description

There are many “special” folders that both Operating Systems employ to facilitate various tasks or features. These include desktop folders, trash folders, preferences folders, and many others.

folderType is a string that specifies which special folder’s path you want. This string is case-insensitive.

Returns the pathname to the desired folder if sucessful or the empty string “” if an error occurs.

Return Type

String

Macintosh Notes

The following table lists what “folderType” to use to obtain the desired folder path.

folderType	**meaning**
kSystemFolderType	System Folder on boot volume
kDesktopFolderType	Desktop folder
kTrashFolderType	Trash folder
kPrintMonitorDocsFolderType	PrintMonitor Documents folder
kStartupItemsDisabledFolderType	Startup Items (Disabled) folder
kShutdownFolderType	Shutdown Items folder
kShutdownItemsDisabledFolderType	Shutdown Items (Disabled) folder
kAppleMenuFolderType	Apple Menu Items folder
kControlPanelFolderType	Control Panels folder
kControlPanelDisabledFolderType	Control Panels (Disabled) folder
kSystemExtensionDisabledFolderType	System Extensions (Disabled) folder
kExtensionFolderType	Extensions folder
kExtensionDisabledFolderType	Extensions (Disabled) folder
kFontsFolderType	Fonts folder
kPreferencesFolderType	Preferences folder
kTemporaryFolderType	Temporary Items folder
kApplicationsFolderType	Applications folder
kDocumentsFolderType	Documents folder

Windows Notes

Windows is pretty picky about what it will tell you regarding these folders. By having Internet Explorer v5.0 or newer installed on your system you will be able to obtain most of these paths. This is because there is a special system file, Shell32.dll, that gets updated every so often and ships with IE.

Any folder name returned has a '\' character appended.

The following table lists what "folderType" to use and its meaning. This is taken from the Microsoft developer web site description for CSIDL values for the SHGetSpecialFolderPath system call. Its web address is: http://msdn.microsoft.com/library/psdk/shellcc/shell/Functions/CSIDL.htm.

folderType	**meaning**
CSIDL_ALTSTARTUP	File system directory that corresponds to the user's nonlocalized Startup program group.
CSIDL_APPDATA	File system directory that serves as a common repository for application-specific data. A typical path is C:\Documents and Settings*username*\Application Data.
CSIDL_COMMON_ALTSTARTUP	(NT only) File system directory that corresponds to the nonlocalized Startup program group for all users.
CSIDL_COMMON_APPDATA	Application data for all users. A typical path is C:\Documents and Settings\All Users\Application Data.
CSIDL_COMMON_DESKTOPDIRECTORY	(NT only) File system directory that contains files and folders that appear on the desktop for all users. A typical path is C:\Documents and Settings\All Users\Desktop.
CSIDL_COMMON_FAVORITES	(NT only) File system directory that serves as a common repository for all users' favorite items.
CSIDL_COMMON_PROGRAMS	(NT only) File system directory that contains the directories for the common program groups that appear on the Start menu for all users. A typical path is C:\Documents and Settings\All Users\Start Menu\Programs.
CSIDL_COMMON_STARTMENU	(NT only) File system directory that contains the programs and folders that appear on the Start menu for all users. A typical path is C:\Documents and Settings\All Users\Start Menu.
CSIDL_COMMON_STARTUP	(NT only) File system directory that contains the programs that appear in the Startup folder for all users. A typical path is C:\Documents and Settings\All Users\Start Menu\Programs\Startup.
CSIDL_COOKIES	File system directory that serves as a common repository for Internet cookies. A typical path

	is C:\Documents and Settings*username*\Cookies.
CSIDL_DESKTOPDIRECTORY	File system directory used to physically store file objects on the desktop (not to be confused with the desktop folder itself). A typical path is C:\Documents and Settings*username*\Desktop.
CSIDL_FAVORITES	File system directory that serves as a common repository for the user's favorite items. A typical path is C:\Documents and Settings*username*\Favorites.
CSIDL_HISTORY	File system directory that serves as a common repository for Internet history items.
CSIDL_INTERNET_CACHE	File system directory that serves as a common repository for temporary Internet files. A typical path is C:\Documents and Settings*username*\Temporary Internet Files.
CSIDL_LOCAL_APPDATA	File system directory that serves as a data repository for local (non-roaming) applications. A typical path is C:\Documents and Settings*username*\Local Settings\Application Data.
CSIDL_MYPICTURES	My Pictures folder. A typical path is C:\Documents and Settings*username*\My Documents\My Pictures.
CSIDL_NETHOOD	A file system folder containing the link objects that may exist in the My Network Places virtual folder. It is not the same as CSIDL_NETWORK, which represents the network namespace root. A typical path is C:\Documents and Settings*username*\NetHood.
CSIDL_PERSONAL	File system directory that serves as a common repository for documents. A typical path is C:\Documents and Settings*username*\My Documents.
CSIDL_PRINTHOOD	File system directory that contains the link objects that may exist in the Printers virtual folder. A typical path is C:\Documents and Settings*username*\PrintHood.
CSIDL_PROFILE	User's profile folder.
CSIDL_PROGRAM_FILES	Program Files folder. A typical path is C:\Program Files.
CSIDL_PROGRAMS	File system directory that contains the user's program groups (which are also file system directories). A typical path is C:\Documents and Settings*username*\Start Menu\Programs.
CSIDL_RECENT	File system directory that contains the user's most recently used documents. A typical path is C:\Documents and Settings\username\Recent.

CSIDL_SENDTO	File system directory that contains Send To menu items. A typical path is C:\Documents and Settings*username*\SendTo.
CSIDL_STARTMENU	File system directory containing Start menu items. A typical path is C:\Documents and Settings*username*\Start Menu.
CSIDL_STARTUP	File system directory that corresponds to the user's Startup program group. The system starts these programs whenever any user logs onto Windows NT or starts Windows 95. A typical path is C:\Documents and Settings*username*\Start Menu\Programs\Startup.
CSIDL_SYSTEM	System folder. A typical path is C:\WINNT\System32.
CSIDL_TEMPLATES	File system directory that serves as a common repository for document templates.
CSIDL_WINDOWS	Windows directory or SYSROOT. This corresponds to the %windir% or %SystemRoot% environment variables. A typical path is C:\WINNT.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FolderGetSpecialPath("kSystemFolderType")
-- "Boot:System Folder:"
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-30	SHGetSpecialFolderLocation() call failed (bummer)	Win
-140	Special folder type specified is unknown	Mac
-141	FindFolder() system call failed	Mac

fx_FolderExists

Name

fx_FolderExists – tell whether a given folder exists

Synopsis

intVar = fx_FolderExists(object me, string folderName)

Description

Use this method to determine the existence of the ***folderName*** path.

Returns True (1) if the folder path exists, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FolderExists("Macintosh HD:System Folder")
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-15	Folder not found	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win

fx_FolderCreate

Name

fx_FolderCreate – create a folder

Synopsis

intVar = fx_FolderCreate(object me, string folderName)

Description

This function creates a folder (directory) with the given path. If a folder or file already exists with the given path and name, the function returns False (0). True (1) is returned if successful.

Specify the folder to create by giving a complete path in ***folderName***.

Note that if you specify multiple levels of folders to create, the method will attempt to create them one at a time until it creates the final folder you specified.

False (0) is returned if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_FolderExists(“\\LinuxBox\kkersten\one”)
-- 0
put fxObj.fx_FolderCreate(“\\LinuxBox\kkersten\one\two\three”)
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-15	Folder not found	both
-16	Specified folder is actually a file	both
-17	Folder creation failure	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win

fx_FolderDelete

Name

fx_FolderDelete – delete a folder

Synopsis

intVar = fx_FolderDelete(object me, string folderName, Boolean recursive)

Description

This method is both powerful and dangerous. It will perform a "tree walk" to delete not only the target directory and all its files given in the ***folderName*** argument, but it will also delete all subfolders and their files if ***recursive*** is set to True.

Note that it is possible to erase an entire hard drive with the careless use of this command! Issuing a command such as fx_FolderDelete(fxObj, "C:\", True) will in fact erase drive C:! Be warned!

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FolderDelete("C:\add to trash")
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-5	File deletion failure	both
-7	File not found	both
-8	Specified file is actually a folder	both
-15	Folder not found	both
-16	Specified folder is actually a file	both
-18	Could not delete specified folder	both
-19	Could not retrieve directory ID number	Mac
-51	Specified volume does not exist	both

fx_FolderRecycle

Name

fx_FolderRecycle – move a folder and its contents to the Trash/Recycle Bin

Synopsis

intVar = fx_FolderRecycle(object me, string folderName)

Description

If you wish to move ***folderName*** to the trash rather than immediately deleting it, use this method.

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FolderRecycle("C:\add to trash")
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-1	General error of unknown origin	both
-15	Folder not found	both
-16	Specified folder is actually a file	both
-20	I/O error	Mac
-21	Hardware volume lock	Mac
-22	Software volume lock	Mac
-23	Target directory is locked	Mac
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win
-95	Attempt to move into offspring	Mac
-98	FolderRecycle failed	Win
-140	Special folder type specified is unknown	Mac
-141	FindFolder() system call failed	Mac

fx_FolderCopy

Name

fx_FolderCopy – copy the contents of a folder and (possibly) subfolders

Synopsis

intVar = fx_FolderCopy(object me, string fromFolderNm, toFolderNm, Boolean recursive)

Description

This method copies all files in ***fromFolderNm*** into ***toFolderNm***. If the ***recursive*** flag is set all subfolders and their files will also be copied.

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

Invisible files are not copied by this or other file or folder commands.

Windows Notes

None.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FolderCopy("Macintosh HD:Documents",
                    "Linux Server:kkersten:Documents Backup", True)
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-1	General error of unknown origin	both
-5	File deletion failure	both
-7	File not found	both
-8	Specified file is actually a folder	both
-9	File creation failure	both
-10	File open failure	Mac
-11	File write failure	Mac
-13	File read failure	Mac
-15	Folder not found	both
-16	Specified folder is actually a file	both
-17	Folder creation failure	both
-19	Could not retrieve directory ID number	Mac
-26	Cannot copy a file onto itself	both
-40	Could not allocate memory for file copy	Mac
-51	Specified volume does not exist	both

fx_FolderMove

Name

fx_FolderMove – move a folder to a new location

Synopsis

intVar = fx_FolderMove(object me, string fromFolderName, string toFolderName)

Description

Use this method to move the contents of a folder, its files and subfolders to a new location within the same volume.

fromFolderName refers to the source folder path to move.

toFolderName refers to the destination folder path.

Note that a folder can only be moved within its current volume. You cannot move a folder to another volume (drive).

It is also impossible to move a folder into one of its child folders.

You cannot move the folder if a folder with the same name already exists at the destination.

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_FolderMove(“C:\First Run”, “C:\My Documents\First Run”)
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-1	General error of unknown origin	both
-15	Folder not found	both
-16	Specified folder is actually a file	both
-20	I/O error	Mac
-21	Hardware volume lock	Mac
-22	Software volume lock	Mac
-23	Target directory is locked	Mac
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win
-95	Attempt to move into offspring	Mac
-96	Destination folder already exists	both
-97	FolderMove failed	Win

fx_FolderSyncOneWay

Name

fx_FolderSyncOneWay – synchronize two folders in one direction

Synopsis

intVar = fx_FolderSyncOneWay(object me, string fromFolderName, string toFolderName, Boolean recursive, Boolean deleteStrays)

Description

This method has some very specialized uses. It "synchronizes" the contents of two folders, but only in one direction, source -> destination. It optionally operates recursively. Another option will clean up the destination directory by deleting files that are not present in the source directory.

Every file in each folder is compared using fx_FileExists and fx_FileCompare. If a file exists in only the source folder, it is copied to the destination folder. If a file exists in both folders, and if the modified date of the file in the source folder is newer than the corresponding file in the destination folder, then the file from the source folder is copied to the destination folder.

In other words, this call ensures that all files in the source folder are the same or newer than the ones in the destination folder.

A "stray" file is one that exists in the destination folder but not in the source folder.

If any of the destination folders do not exist, they will be created.

fromFolderName refers to the source or controlling folder.

toFolderName refers to the destination folder.

If the ***recursive*** flag is set to True, then the command will operate on all subfolders found.

If you want all files removed from the destination directory that do not exist in the source directory, set the ***deleteStrays*** flag to True.

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FolderSyncOneWay("Boot:Always Current:", "Macintosh HD:My Files",
                              True, True)
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-1	General error of unknown origin	both
-5	File deletion failure	both
-7	File not found	both
-8	Specified file is actually a folder	both
-9	File creation failure	both
-10	File open failure	Mac
-11	File write failure	Mac
-13	File read failure	Mac
-15	Folder not found	both
-16	Specified folder is actually a file	both
-17	Folder creation failure	both
-19	Could not retrieve directory ID number	Mac
-26	Cannot copy a file onto itself	both
-40	Could not allocate memory for file copy	Mac
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win

fx_FolderSyncBothWays

Name

fx_FolderSyncBothWays – synchronize two folders in both directions

Synopsis

intVar = fx_FolderSyncBothWays(object me, string folderName1, string folderName2,
Boolean recursive)

Description

This is another very specialized method. It synchronizes the contents of two folders in both directions. It optionally operates recursively.

Every file in each folder is compared using fx_FileExists and fx_FileCompare. If a file exists in only one folder, it is copied to the other folder. If a file exists in both folders, then the newest version of the file, determined by its modification date, is copied to the other folder.

folderName1 refers to the first folder.

folderName2 refers to the second folder.

If the ***recursive*** flag is set to True, then the command will operate on all subfolders found.

Returns True (1) if successful, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_FolderSyncBothWays(“Boot:Always Current:”, “Macintosh HD:My Files”, True)
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-1	General error of unknown origin	both
-5	File deletion failure	both
-7	File not found	both
-8	Specified file is actually a folder	both
-9	File creation failure	both
-10	File open failure	Mac
-11	File write failure	Mac
-13	File read failure	Mac
-15	Folder not found	both
-16	Specified folder is actually a file	both
-17	Folder creation failure	both
-19	Could not retrieve directory ID number	Mac
-26	Cannot copy a file onto itself	both
-40	Could not allocate memory for file copy	Mac
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win

fx_FolderToList

Name

fx_FolderToList – return a list of folders given a path

Synopsis

listVar = fx_FolderToList(object me, string folderName)

Description

This function will create a Director list and place an item in the list for each file and folder found within the given folder name.

folderName refers to the folder to create a list from.

If an error occurs, an empty list [ ] is returned.

Return Type

List

Macintosh Notes

Folder names in the list have a `:' character appended.

Windows Notes

Folder names in the list have a `\' character appended.

Note that on Windows you get back a listing of the directory's contents, which is probably not sorted. You may want to perform a sort on the list before using it.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FolderToList("c:\temp")
-- ["abc.txt", "AtGuard\", "author.ppt", "babbling.doc", "matrix.exe"]
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-15	Folder not found	both
-19	Could not retrieve directory ID number	Mac

Volume Methods

These are the methods that operate on volumes. A volume can be an entire disk drive or it can be a “mount point” on a network file server.

fx_VolumeExists
fx_VolumeGetFreeBytes
fx_VolumeGetTotalBytes
fx_VolumeIsCDROM
fx_VolumeIsRemovable
fx_VolumeEject
fx_VolumesToList

fx_VolumeExists

Name

fx_VolumeExists – determine if the named volume exists on the system

Synopsis

intVar = fx_VolumeExists(object me, string volumeName)

Description

Returns True (1) if ***volumeName*** exists, False (0) if not, or if an error occurs.

Return Type

Integer

Macintosh Notes

On the Macintosh it is possible to have more than one volume with the same name. Be aware that FileXtra3 cannot reconcile this, and it will find the volume with the correct name that was mounted first.

Windows Notes

Note that you can pass in a complete path to this call and it won't care. All it is interested in is the volume name; however it makes no attempt to verify the validity of any path passed in as an argument.

Also note that the Windows system call expects at a minimum a path to the root directory on a volume, such as "c:\". Using "c:" would cause the method to fail.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_VolumeExists("\\LinuxBox\\kkersten\")
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win

fx_VolumeGetFreeBytes

Name

fx_VolumeGetFreeBytes – return the amount of free space on a volume

Synopsis

floatVar = fx_VolumeGetFreeBytes(object me, string volumeName)

Description

Call this method to determine the number of bytes available on ***volumeName***.

False (0) is returned on error.

Return Type

Float

Macintosh Notes

None.

Windows Notes

If you are using Windows 95 you must be using OSR2 and have Internet Explorer 4.0 or newer installed for this method to work properly.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_VolumeGetFreeBytes("C:\")
-- 14947409920.0000
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-15	Folder not found	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win

fx_VolumeGetTotalBytes

Name

fx_VolumeGetTotalBytes – return the total size of a volume

Synopsis

floatVar = fx_VolumeGetTotalBytes(object me, string volumeName)

Description

Call this method to determine the total number of bytes on ***volumeName***.

False (0) is returned on error.

Return Type

Float

Macintosh Notes

None.

Windows Notes

If you are using Windows 95 you must be using OSR2 and have Internet Explorer 4.0 or newer installed for this method to work properly.

Example

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_VolumeGetTotalBytes(“C:\”)
-- 20415111168.0000
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-15	Folder not found	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win

fx_VolumeIsCDROM

Name

fx_VolumeIsCDROM – determine if a volume is a CD-ROM drive

Synopsis

intVar = fx_VolumeIsCDROM(object me, string volumeName)

Description

Call this method to determine if ***volumeName*** is a CD-ROM drive.

True (1) is returned if the volume is a CD-ROM drive, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

This function checks the “lock” bit for the requested volume. This could also occur in odd circumstances for volumes other than CD-ROM drives, so you could double-check the results with fx_VolumeIsRemovable().

Windows Notes

None.

Example

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_VolumeIsCDROM(“My Burned Files”)
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-15	Folder not found	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win
-61	Specified volume is not a CD-ROM	both

fx_VolumeIsRemovable

Name

fx_VolumeIsRemovable – determine if a volume uses removable media

Synopsis

intVar = fx_VolumeIsRemovable(object me, string volumeName)

Description

Call this method to determine if ***volumeName*** uses removable media. This applies to CD-ROM drives as well.

Returns True (1) if volume supports removable media, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra(“FileXtra3”).new()
put fxObj.fx_VolumeIsRemovable(“My Burned Files”)
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-62	Specified volume is not removable	both

fx_VolumeEject

Name

fx_VolumeEject – eject media from a volume

Synopsis

intVar = fx_VolumeEject(object me, string volumeName)

Description

Call this method to eject media from a drive that supports removable media.

volumeName refers to the volume to eject media from.

Returns True (1) if media is successfully ejected, False (0) if not or if an error occurs.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_VolumeEject("My Burned Files")
-- 1
fxObj = 0
```

Error Codes

Code	Message	Platform
0	Successful completion	both
-16	Specified folder is actually a file	both
-51	Specified volume does not exist	both
-62	Specified volume is not removable	both
-63	Specified volume has open files on it	Mac
-64	Problems with Eject() call	Mac
-65	Problems with UnmountVol() call	Mac
-81	Specified application was not found; process not created	Win
-82	Cannot unlock media	Win
-83	Cannot eject media	Win
-84	Cannot eject volume	Win

fx_VolumesToList

Name

fx_VolumesToList – return a list of volumes on the system

Synopsis

listVar = fx_VolumesToList(object me)

Description

Call this method to obtain a list of volumes. Note that volumes that support removable media but do not currently have media inserted are still listed. You can call fx_VolumeExists followed by fx_ErrorNumber to check a volume and see if media is mounted or not.

Returns the empty list [ ] if an error occurs.

Return Type

List

Macintosh Notes

Trailing ':' characters are appended to the names listed. The Mac OS always returns the directory list in alphabetical order.

Windows Notes

Trailing '\' characters are appended to the names listed.

Note that no UNC-named volumes will be listed with this call. If any remotely mounted volumes have been mapped to drive letters, then those drive letters appear in the list.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_VolumesToList()
-- ["Boot:", "X:", "Kent 30gb:", "Public Files:"]
fxObj = 0
```

Error Codes

None.

Error Reporting Methods

These two methods report the error code of the last method to be invoked and provide a text string interpreting what the error means or what happened to cause the error.

fx_ErrorNumber
fx_ErrorString

fx_ErrorNumber

Name

fx_ErrorNumber – return the error code from the most recent method call

Synopsis

intVar = fx_ErrorNumber(object me)

Description

Each time you call a FileXtra3 method and it returns False, you should call fx_ErrorNumber to determine the error code. Your lingo code can then decide best how to recover or proceed.

Return Type

Integer

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileExists("c:\brownwood derby.htm")
-- 0
put fxObj.fx_ErrorNumber()
-- -7
fxObj = 0
```

Error Codes

None.

fx_ErrorString

Name

fx_ErrorString – return an error string from the most recent method call

Synopsis

stringVar = fx_ErrorString(object me)

Description

Each time you call a FileXtra3 method and it returns False, you can call fx_ErrorString to receive a human-readable explanation of what happened. This won't always be useful information to you, such as when a link file on Windows can't be created and you get back a message that says:

SetPath system call failed

But it will give you enough information (hopefully) to contact me and pass it along in case it is a bug.

If the error code returned by the previous method call is undefined (which is very unlikely), you will see the following message:

Undetermined error

If the error code returned by the previous method indicates a successful completion (True return code), the message you will see is:

Successful completion

Return Type

String

Macintosh Notes

None.

Windows Notes

None.

Example

```
fxObj = xtra("FileXtra3").new()
put fxObj.fx_FileExists("c:\brownwood derby.htm")
-- 0
put fxObj.fx_ErrorString()
-- "File not found"
fxObj = 0
```

Error Codes

None.

Appendix A. Complete Listing of Error Codes

Code	Message	Platform
0	Successful completion	both
-1	General error of unknown origin	both
-5	File deletion failure	both
-6	File rename failure	Mac
-7	File not found	both
-8	Specified file is actually a folder	both
-9	File creation failure	both
-10	File open failure	Mac
-11	File write failure	Mac
-13	File read failure	Mac
-14	Destination volume full	Mac
-15	Folder not found	both
-16	Specified folder is actually a file	both
-17	Folder creation failure	both
-18	Could not delete specified folder	both
-19	Could not retrieve directory ID number	Mac
-20	I/O error	Mac
-21	Hardware volume lock	Mac
-22	Software volume lock	Mac
-23	Target directory is locked	Mac
-26	Cannot copy a file onto itself	both
-30	SHGetSpecialFolderLocation() call failed (bummer)	Win
-40	Could not allocate memory for file copy	Mac
-42	Could not obtain Finder information for file	Mac
-44	Not enough memory to launch application	Mac
-51	Specified volume does not exist	both
-52	Specified volume exists but is not mounted	Win
-56	SHBrowseForFolder failed	Win
-57	SHGetPathFromIDList failed	Win
-61	Specified volume is not a CD-ROM	both
-62	Specified volume is not removable	both
-63	Specified volume has open files on it	Mac
-64	Problems with Eject() call	Mac
-65	Problems with UnmountVol() call	Mac
-71	No file type found for specified file	Win
-73	No application associated with specified file type	Win
-74	No \\shell\\open\\command key found for specified file type	Win
-75	No \\shell\\print\\command key found for specified file type	Win
-77	Problems reading desktop database	Mac
-81	Specified application was not found; process not created	Win
-82	Cannot unlock media	Win
-83	Cannot eject media	Win
-84	Cannot eject volume	Win
-91	Destination file already exists	both
-92	FileMove failed	Win
-93	FileRecycle failed	Win
-95	Attempt to move into offspring	Mac
-96	Destination folder already exists	both
-97	FolderMove failed	Win
-98	FolderRecycle failed	Win
-101	File sizes are different	both
-103	File two’s mod date is newer than file one’s	both

-105	File one’s mod date is newer than file two’s	both
-122	Could not create FSSpec record	Mac
-123	Could not create FSSpec record	Mac
-124	NewAlias() toolbox call failed	Mac
-125	NewAlias() toolbox call returned nil	Mac
-126	Creating resource fork of alias file failed	Mac
-127	Opening resource fork of alias file failed	Mac
-128	AddResource() on alias file failed	Mac
-129	WriteResource() on alias file failed	Mac
-130	CloseResFile() on alias file failed	Mac
-140	Special folder type specified is unknown	Mac
-141	FindFolder() system call failed	Mac
-150	Specified link file is actually a normal file	both
-152	Could not read the alias resource	Mac
-154	ResolveAlias() failed	Mac
-155	Could not resolve alias path	Win
-161	SetPath system call failed	Win
-162	SetDescription system call failed	Win
-163	IPersistFile::Save system call failed	Win
-210	New filename already exists or two paths are different	Win