3 File System Access

Adobe ExtendScript defines classes that simplify cross-platform file-system access. These classes are available to all applications that support a JavaScript interface.

- The first part of this chapter, Using File and Folder objects, describes how to use these classes and provides details of pathname syntax.
- "File object" on page 47 and "Folder object" on page 56 provide reference details of the objects, properties, methods, and creation parameters. You can also choose the Core JavaScript Classes dictionary from the Help menu in the ExtendScript Toolkit to inspect the objects in the Object Model Viewer.

Using File and Folder objects

Because path name syntax is very different on Windows, Mac OS, and UNIX®, Adobe ExendScript defines the `File` and `Folder` objects to provide platform-independent access to the underlying file system. A `File` object represents a disk file, a `Folder` object represents a directory or folder.

- The `Folder` object supports file system functionality such as traversing the hierarchy; creating, renaming or removing files; or resolving file aliases.
- The `File` object supports input/output functions to read or write files.

There are several ways to distinguish between a `File` and a `Folder` object. For example:

```
if (f instanceof File) ...

if (typeof f.open == "undefined") ...// Folders do not open
```

`File` and `Folder` objects can be used anywhere that a path name is required, such as in properties and arguments for files and folders.

NOTE: When you create two `File` objects that refer to the same disk file, they are treated as distinct objects. If you open one of them for I/O, the operating system may inhibit access from the other object, because the disk file already is open.

Specifying paths

When creating a `File` or `Folder` object, you can specify a platform-specific path name, or an absolute or relative path in a platform-independent format known as *universal resource identifier* (*URI*) notation. The path stored in the object is always an absolute, full path name that points to a fixed location on the disk.

- Use the `toString` method to obtain the name of the file or folder as string containing an absolute path name in URI notation.
- Use the `fsName` property to obtain the platform-specific file name.

Absolute and relative path names

An absolute path name in URI notation describes the full path from a root directory down to a specific file or folder. It starts with one or two slashes (/), and a slash separates path elements. For example, the following describes an absolute location for the file myFile.jsx:

```
/dir1/dir2/mydir/myFile.jsx
```

A relative path name in URI notation is appended to the path of the current directory, as stored in the globally available current property of the Folder class. It starts with a folder or file name, or with one of the special names dot (.) for the current directory, or dot dot (..) for the parent of the current directory. A slash (/) separates path elements. For example, the following paths describe various relative locations for the file myFile.jsx:

myFile.jsx ./myFile.jsx	In the current directory.
../myFile.jsx	In the parent of the current directory.
../../myFile.jsx	In the grandparent of the current directory.
../dir1/myFile.jsx	In dir1, which is parallel to the current directory.

Relative path names are independent of different volume names on different machines and operating systems, and therefore make your code considerably more portable. You can, for example, use an absolute path for a single operation, to set the current directory in the Folder.current property, and use relative paths for all other operations. You would then need only a single code change to update to a new platform or file location.

Character interpretation in paths

There are some platform differences in how pathnames are interpreted:

- On Windows and Mac OS, path names are not case sensitive. In UNIX, paths are case sensitive.
- On Windows, both the slash (/) and the backslash (\) are valid path element separators. Backslash is the escape character, so you must use a double backslash (\\) to indicate the character.
- On Mac OS, both the slash (/) and the colon (:) are valid path element separators.

If a path name starts with two slashes (or backslashes on Windows), the first element refers to a remote server. For example, //myhost/mydir/myfile refers to the path /mydir/myfile on the server myhost.

URI notation allows special characters in pathnames, but they must specified with an escape character (%) followed by a hexadecimal character code. Special characters are those which are not alphanumeric and not one of the characters:

```
/ - — . ! ~ * ' ( )
```

A space, for example, is encoded as %20, so the file name "my file" is specified as "my%20file". Similarly, the character ä is encoded as %E4, so the file name "Bräun" is specified as "Br%E4un".

This encoding scheme is compatible with the global JavaScript functions encodeURI and decodeURI.

The home directory

A path name can start with a tilde (~) to indicate the user's home directory. It corresponds to the platform's `HOME` environment variable.

UNIX and Mac OS assign the `HOME` environment variable according to the user login. On Mac OS, the default home directory is `/Users/`*`username`*. In UNIX, it is typically `/home/`*`username`* or `/users/`*`username`*. ExtendScript assigns the home directory value directly from the platform value.

On Windows, the `HOME` environment variable is optional. If it is assigned, its value must be a Windows path name or a path name referring to a remote server (such as `\\myhost\mydir`). If the `HOME` environment variable is undefined, the ExtendScript default is the user's home directory, usually the `C:\Documents and Settings\`*`username`* folder.

NOTE: A script can access many of the folders that are specified with platform-specific variables through static, globally available `Folder` class properties; for instance, `appData` contains the folder that stores application data for all users.

Volume and drive names

A volume or drive name can be the first part of an absolute path in URI notation. The values are interpreted according to the platform.

Mac OS volumes

When Mac OS X starts, the startup volume is the root directory of the file system. All other volumes, including remote volumes, are part of the `/Volumes` directory. The `File` and `Folder` objects use these rules to interpret the first element of a path name:

- If the name is the name of the startup volume, discard it.
- If the name is a volume name, prepend `/Volumes`.
- Otherwise, leave the path as is.

Mac OS 9 is not supported as an operating system, but the use of the colon as a path separator is still supported and corresponds to URI and to Mac OS X paths as shown in the following table. These examples assume that the startup volume is `MacOSX`, and that there is a mounted volume `Remote`.

URI path name	Mac OS 9 path name	Mac OS X path name
`/MacOSX/dir/file`	`MacOSX:dir:file`	`/dir/file`
`/Remote/dir/file`	`Remote:dir:file`	`/Volumes/Remote/dir/file`
`/root/dir/file`	`Root:dir:file`	`/root/dir/file`
`~/dir/file`		`/Users/jdoe/dir/file`

Windows drives

On Windows, volume names correspond to drive letters. The URI path `/c/temp/file` normally translates to the Windows path `C:\temp\file`.

If a drive exists with a name matching the first part of the path, that part is always interpreted as that drive. It is possible for there to be a folder in the root that has the same name as the drive; imagine, for example,