AppleSingle/AppleDouble Formats for Foreign Files Developer’s Note

 Apple Computer, Inc.

Preface

This document describes version 2 of the AppleSingle/AppleDouble file formats.

To compare this version with version 1 of the AppleSingle/AppleDouble file formats, refer to Appendix B of the *A/UX Toolbox: Macintosh ROM Interface* manual.

You can also find a description of version 1 in the Apple II File Type Notes for file types $E0.0001-AppleSingle and $E0.0002/$E0.0003-AppleDouble. These File Type Notes are available through Apple Developer Technical Support.

Chapter 1:
About AppleSingle/AppleDouble Formats

When a file is exchanged between file systems that do not support the same file attributes, some of the file's attributes are lost. Apple Computer has developed two file formats, called **AppleSingle** and **AppleDouble**, that allow files to maintain their file attributes on foreign file systems that do not support the same attributes.

The AppleSingle and AppleDouble formats were initially developed to store Macintosh® files on file systems that do not support the Macintosh file structure. However, these formats can also be used to represent almost any kind of file on almost any file system. They assume only that the file systems used allow you to create a file as a set of contiguous bytes.

The AppleSingle/AppleDouble file format is not tied to a single home file system. A file is stored as a heterogeneous collection of data and attributes, which are interpreted as needed by the application reading the file.

The AppleSingle or AppleDouble format can be used

- as a standard format for transferring files among differing, or heterogeneous, computers.
- as a standard format for transferring files within a single computer.

Choosing AppleSingle or AppleDouble

The AppleSingle and AppleDouble formats are alike in that they use the same components to represent a file on a foreign system: data, resources, and attributes. The difference between the two formats is that the AppleSingle format stores these components in a single foreign file, and the AppleDouble format stores these components in two foreign files—one for the data, the other for the resources and attributes.

Applications may use either AppleSingle or AppleDouble when they create files on foreign file systems; however, they must understand both formats.

To find out which format is appropriate for you, examine your application. If the AppleSingle format is used, home file system data cannot be moved or deleted inconsistently; however, the AppleSingle format is harder to update. The AppleDouble format is easier to modify but one of the two AppleDouble files can be moved or deleted inconsistently.

If you are building an AppleTalk Filing Protocol (AFP) server (or any other server that supports the Macintosh computer), then you may want to use one of these formats as your application’s internal or external file storage format. The choice is yours. It doesn’t matter which format is used within your application, but it is suggested that you use one of these formats as your external storage format so that files can be shared by other applications running on the same machine as your application.

Terminology

The following terms are relevant to the discussion of AppleSingle and AppleDouble file formats.

A Macintosh file has two forks: the resource fork and the data fork. The **resource fork** contains data used by an application, such as menus, fonts, and icons. An executable file's code is also stored in the resource fork. The **data fork** contains data specific to an application.

The **home file system** is the primary file system for which the file's contents were created. The home file system is not necessarily the file system in which the file was created. For example, if a program running on a UNIX® system creates a file that holds a MacWrite® document, the file's home file system is the Macintosh file system—not the UNIX file system—because the file's contents are formatted for a Macintosh application.

In contrast to the home file system is the **foreign file system**, which is the other file system that stores or processes the created file. In the previous example, the UNIX file system is the foreign file system.

Chapter 2:
The AppleSingle Format

In the AppleSingle format, a file's contents and attributes are stored in a single file in the foreign file system. For example, both forks of a Macintosh file, the Finder™ information, and an associated comment are arranged in a single file with a simple structure.

An AppleSingle file consists of a header followed by one or more data entries. The header consists of several fixed fields and a list of entry descriptors, each pointing to a data entry. Each entry is optional and may or may not appear in the file.

AppleSingle file header

Table 2-1 describes the contents of an AppleSingle file header.

Table 2-1 AppleSingle file header

Field	Length
Magic number	4 bytes
Version number	4 bytes
Filler	16 bytes
Number of entries	2 bytes
Entry descriptor for each entry:	
Entry ID	4 bytes
Offset	4 bytes
Length	4 bytes

Byte ordering in the file header fields follows MC68000 conventions, most significant byte first. The fields in the header file follow the conventions described in the following sections.

Magic number

This field, modeled after the UNIX magic number feature, specifies the file's format. Apple has defined the magic number for the AppleSingle format as $00051600 or 0x00051600.

Version number

This field denotes the version of AppleSingle format in the event the format evolves (more fields may be added to the header). The version described in this developer's note is version $00020000 or 0x00020000.

Filler

This field is all zeros ($00 or 0x00).

Number of entries

This field specifies how many different entries are included in the file. It is an unsigned 16-bit number. If the number of entries is any number other than 0, then that number of entry descriptors immediately follows the number of entries field.

Entry descriptors

The entry descriptors are made up of the following three fields:

- **Entry ID,** an unsigned 32-bit number, defines what the entry is. Entry IDs range from 1 to $FFFFFFFF. Entry ID 0 is invalid.

- **Offset,** an unsigned 32-bit number, shows the offset from the beginning of the file to the beginning of the entry's data.

- **Length,** an unsigned 32-bit number, shows the length of the data in bytes. The length can be 0.

Predefined entry IDs

Apple has defined a set of entry IDs and their values as follows:

Data Fork	1	Data fork
Resource Fork	2	Resource fork
Real Name	3	File's name as created on home file system
Comment	4	Standard Macintosh comment
Icon, B&W	5	Standard Macintosh black and white icon
Icon, Color	6	Macintosh color icon
File Dates Info	8	File creation date, modification date, and so on
Finder Info	9	Standard Macintosh Finder information
Macintosh File Info	10	Macintosh file information, attributes, and so on
ProDOS File Info	11	ProDOS file information, attributes, and so on
MS-DOS File Info	12	MS-DOS file information, attributes, and so on
Short Name	13	AFP short name
AFP File Info	14	AFP file information, attributes, and so on
Directory ID	15	AFP directory ID

Apple reserves the range of entry IDs from 1 to $7FFFFFFF. The rest of the range is available for applications to define their own entries. Apple does not arbitrate the use of the rest of the range.

Applications reading an AppleSingle or AppleDouble file interpret the entry IDs that are relevant to their home file system and treat the other entries as opaque data. For example, a Macintosh client application will understand entry IDs 1 through 6 and 8 through 10. A ProDOS® client application reading the same file will read entry ID 11 instead of entry IDs 9 and 10.

Entry IDs 1, 3, and 8 are typically created for all files; entry ID 2 only for Macintosh and ProDOS files; entry IDs 4, 5, 6, 9, and 10 only for Macintosh files; entry ID 11 only for ProDOS files; entry ID 12 only for MS-DOS files; and entry IDs 13, 14, and 15 only by AFP servers.

Macintosh Icon entries do not appear in most files because they are typically stored as a bundle in the application file's resource fork.

The File Dates Info entry (ID=8) consists of the file creation, modification, backup and access times (see Figure 2-1), stored as a signed number of seconds before or after 12:00 a.m. (midnight), January 1, 2000 Greenwich Mean Time (GMT). In other words, the start of the year 2000 GMT corresponds to a date-time of 0. Applications must convert to their native date and time conventions. When initially created, a file's backup time and any unknown entries are set to $80000000 or 0x80000000, the earliest reasonable time.

Figure 2-1 File Dates Info

The Macintosh Finder Info entry (ID=9) consists of 16 bytes of Finder Info followed by 16 bytes of extended Finder Info; that is, the field `ioFlFndrInfo` followed by `ioFlXFndrInfo`, as returned by the Macintosh `PBGetCatInfo` call. (The `PBGetCatInfo` and the internal structures of `ioFlFndrInfo` and `ioFlXFndrInfo` are described in *Inside Macintosh*.)

Newly created files have 0s in all Finder Info subfields. If you are creating an AppleSingle or AppleDouble file, you may assign 0 to any subfield whose value

is unknown (most subfields are undefined if the file does not reside on a valid hierarchical file system [HFS] volume), but you may want to set the `fdType` and `fdCreator` subfields.

One additional field must be set in the directory Finder Info entry. Whenever the Finder encounters a new (“inited” bit clear) directory, it initializes the `frView` field of the directory Finder Info to a value indicating how the contents of the directory should be viewed when opened (by icon, by small icon, and so on). Zero is not a legal value. This is not a problem as long as the client has write permission for the directory.

The Finder will place a legal value into the `frView` field when it initializes the directory. However, if the directory permissions deny making changes, the initial value will remain unchanged and cause the Finder to display things strangely. Consequently, it is appropriate to set the `frView` field when the Finder Info is assigned to a new host directory. For example:

```
#define closedView 256   /* Normal view - icon, spatial */
/* ... */
dirFinderInfo.frView = closedView;
```

This results in the traditional “view by icon” display when the Finder first opens the window for that directory.

The Macintosh File Info entry (ID=10) is 32 bits that stores the locked and protected bit (see Figure 2-2). Macintosh file times are stored in entry ID 8.

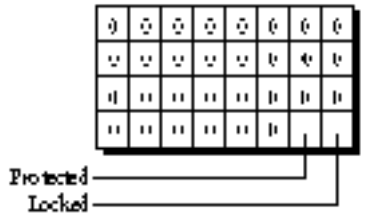

Figure 2-2 Macintosh File Info

The ProDOS File Info entry (ID=11) consists of the file access, file type, and file auxiliary type (see Figure 2-3). ProDOS file times are stored in entry ID 8.

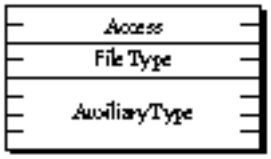

Figure 2-3 ProDOS File Info

The Access word may be used directly in ProDOS 16 or GS/OS calls; only the low byte is significant to ProDOS 8. The File Type word is the file type of the original file; only the low byte is significant to ProDOS 8. The Auxiliary Type long word is the auxiliary type of the original file; only the low word is significant to ProDOS 8.

UNIX file times are stored in entry ID 8.

The MS-DOS File Info entry (ID=12) is 16 bits that stores the MS-DOS attributes (see Figure 2-4). MS-DOS file times are stored in entry ID 8.

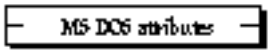

Figure 2-4 MS-DOS File Info

The Short Name entry (ID=13) is the AFP short name. AFP servers must keep a mapping of short names for all foreign files. This short name is stored in entry ID 13.

If entry ID 13 does not exist, an AFP server derives a short name and creates this entry. When they derive short names, AFP servers must ensure that the derived names are unique within the directory. In order to ensure that short names for new foreign files don’t conflict with existing short names, the following convention is used: AFP servers start all derived names with the character “!” ($21 or 0x21). AFP clients are not allowed to access foreign files that begin with this character (!). Except for this restriction on foreign AFP servers, the short name algorithm remains flexible.

The AFP File Info entry (ID=14) is the AFP attributes word, as shown in Figure 2-5. AFP servers should set the BackupNeeded bit whenever a file is modified, or if the AppleSingle/AppleDouble foreign file’s modification time is later than the modification time in entry ID 8.

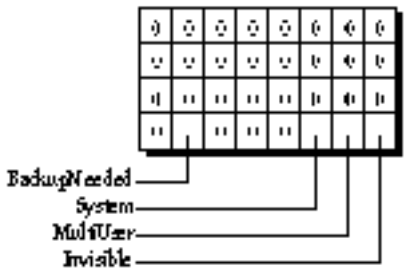

Figure 2-5 AFP File Info

The Directory entry (ID=15) is the AFP directory ID. AFP servers keep the 4-byte directory ID in this entry. This ID is assigned when a directory is created, or when a directory AppleDouble header file without an ID entry is found. The next unused ID is kept in the volume root info file, %RootInfo, in the volume root directory. The %RootInfo file is an AppleDouble header file with entries 3, 4, 8, 9, and 15 (Real Name, Comment, Volume Dates, Finder Info, and Next File ID). It must be locked when it is updated.

File layout

The entry data follows all of the entry descriptors. The data in each entry must be in a single, contiguous block. You can leave holes in the file for later expansion of data. For example, even if a file's comment field is only 10 bytes long, you can place the offset of the next field 200 bytes beyond the offset of the comment field, leaving room for the comment to grow to its maximum length of 200 bytes.

The entries can appear in any order, but you can maximize the efficiency of file access by following these conventions:

- Put the data fork entry at the end of the file. The data fork is the most commonly extended entry, and it is easier to increase its length if it is the last entry in the file.

- Put the entries that are most often read, such as Finder Info, File Dates Info, and Macintosh File Info, as close as possible to the header, to increase the probability that a read of the first block or two will retrieve these entries.

- Allocate the resource fork entry in blocks of 4 kilobytes (K) in order to minimize reorganization of AppleSingle files during updates to the resource fork.

Filename conventions

AppleSingle name derivations for some of the file systems of interest are defined in the following sections.

ProDOS

To generate the AppleSingle filename, use character substitution or deletion to remove illegal characters, and use truncation, if necessary, to reduce the length of the name to the maximum filename length of 15 characters.

MS-DOS

To generate the AppleSingle filename, use character substitution or deletion to remove illegal characters, and use truncation, if necessary, to reduce the length of the name to 8 characters. Then add the MS-DOS extension that is most appropriate to the file (for example, .TXT for a pure text file).

UNIX and NFS

AppleSingle naming conventions for UNIX and the Network File System (NFS) are essentially the same; they vary only according to the capabilities of the foreign file system. The creating application must use the naming convention that is the most complete subset of the foreign file system, and the naming convention must be constant across a single foreign file system volume.

There are three naming conventions:

- 8-bit
- 7-bit ASCII
- 7-bit alphanumeric

With the 8-bit naming convention, the foreign file system can store at least 93 character filenames consisting of all 8-bit characters except slash ($2f or 0x2f), null ($00 or 0x00), and percent ($25 or 0x25). The slash, null, and percent characters are replaced by a percent character, followed by the two-character hexadecimal code of the escaped character. For example, the home file system name *Cañada return - 20%* becomes *Cañada return - 20%25*.

With the 7-bit ASCII naming convention, the foreign file system can store at least 93 character filenames consisting of all 7-bit ASCII characters except slash ($2f or 0x2f), null ($00 or 0x00), and percent ($25 or 0x25). The slash, null, and percent characters are replaced by a percent character, followed by the two-character hexadecimal code of the escaped character. In addition, all extended 8-bit characters ($80 – $ff or 0x80 – 0xff) are replaced by a percent character, followed by the two-character hexadecimal code of the escaped character. For example, the home file system name *Cañada return - 20%* becomes *Ca%96ada return - 20%25*.

With the 7-bit alphanumeric naming convention, the foreign file system can store at least 93 character filenames consisting of all 7-bit alphanumeric characters except slash ($2f or 0x2f), null ($00 or 0x00), and percent ($25 or 0x25). The slash, null, and percent characters are replaced by a percent character, followed by the two-character hexadecimal code of the escaped character. In addition, all nonalphanumeric characters except underscore ($5f or 0x5f) and the last period ($2e or 0x2e) are replaced by a percent character, followed by the two-character hexadecimal code of the escaped character. For example, the home file system name *Cañada return - 20%* becomes *Ca%96ada%20return%20%2d%2020%25*.

Since expanding characters into “%” followed by two hexadecimal digits can result in long filenames, some foreign systems may not be able to store complete 31-character filenames. Behavior under these conditions is not defined here.

Usage

AppleSingle files must be locked during access to ensure data integrity.

Applications that access AppleSingle files ignore all unknown entries, yet preserve the unknown entries when moving or copying files.

The Real Name entry is associated with file storage. File server applications typically use the reverse mapping of the foreign filename when they present names to the client. Servers do not create or read a Real Name entry.

When the home file system name is renamed, the foreign filename must be renamed according to the filename conventions described in the previous sections.

Chapter 3:
The AppleDouble Format

The AppleDouble format uses two files to store data, resources, and attributes. The **AppleDouble Data file** contains the data fork and the **AppleDouble Header file** contains the resource fork.

The AppleDouble Data file contains the standard Macintosh data fork with no additional header. The AppleDouble Header file has exactly the same format as the AppleSingle file, except that it does not contain a data fork entry. The magic number in the AppleDouble Header file differs from the magic number in the AppleSingle Header file so that an application can tell whether it needs to look in another file for the data fork. The magic number for the AppleDouble format is $00051607 or 0x00051607.

The entries in the AppleDouble Header file can appear in any order; however, since the resource fork is the entry that is most commonly extended (after the data fork), Apple recommends that the resource fork entry be placed last in the file. The data fork is easily extended because it resides by itself in the AppleDouble Data file.

It is possible to create a new type of entry in the AppleDouble Header file that points to the AppleDouble Data file to make it easy to find. For example, an application-defined entry might hold the name of the AppleDouble Data file. Some foreign file systems might provide a feature that allows a more permanent pointer to be created—one that would not require the pointer to be updated if the AppleDouble Data file were renamed.

Filename conventions

The following sections present a standard for deriving the AppleDouble Data and AppleDouble Header filenames from the file’s Real Name. Because filename syntax differs in the various file systems, the standard varies by file system.

Knowing the AppleDouble name derivations for some of the file systems of interest will allow applications running on foreign file systems and users to see which files are AppleDouble pairs. Users who know the derivation can rename or move the files while preserving the connection between the two. However, there is no guaranteed way to prevent one file in the pair from being inconsistently renamed, moved, or deleted.

ProDOS

To generate the AppleDouble Data filename, use character substitution or deletion to remove illegal characters, and use truncation, if necessary, to reduce the length of the name to 13 characters, two characters less than the maximum filename length.

To generate the AppleDouble Header filename, prefix the AppleDouble Data filename with the characters uppercase-R period (R.).

MS-DOS

To generate the AppleDouble Data filename, use character substitution or deletion to remove illegal characters, and use truncation, if necessary, to reduce the length of the name to eight characters. Then add the MS-DOS extension that is most appropriate to the file (for example, .TXT for a pure text file).

To generate the AppleDouble Header filename, add the extension .ADF (for AppleDouble file) to the eight-character filename.

UNIX and NFS

AppleSingle/AppleDouble naming conventions for UNIX and NFS are essentially the same; they vary only according to the capabilities of the foreign file system. The creating application must use the naming convention that is the most complete subset of the foreign file system, and the naming convention must be constant across a single foreign file system volume.

There are three naming conventions:

- 8-bit
- 7-bit ASCII

- 7-bit alphanumeric

With the 8-bit naming convention, the foreign file system can store at least 93 character filenames consisting of all 8-bit characters except slash ($2f or 0x2f), null ($00 or 0x00), and percent ($25 or 0x25). The slash, null, and percent characters are replaced by a percent character, followed by the two-character hexadecimal code of the escaped character. For example, the home file system name *Cañada return - 20%* becomes *Cañada return - 20%25*.

With the 7-bit ASCII naming convention, the foreign file system can store at least 93 character filenames consisting of all 7-bit ASCII characters except slash ($2f or 0x2f), null ($00 or 0x00), and percent ($25 or 0x25). The slash, null, and percent characters are replaced by a percent character, followed by the two-character hexadecimal code of the escaped character. In addition, all extended 8-bit characters ($80 – $ff or 0x80 – 0xff) are replaced by a percent character, followed by the two-character hexadecimal code of the escaped character. For example, the home file system name *Cañada return - 20%* becomes *Ca%96ada return - 20%25*.

With the 7-bit alphanumeric naming convention, the foreign file system can store at least 93 character filenames consisting of all 7-bit alphanumeric characters except slash ($2f or 0x2f), null ($00 or 0x00), and percent ($25 or 0x25). The slash, null, and percent characters are replaced by a percent character, followed by the two-character hexadecimal code of the escaped character. In addition, all nonalphanumeric characters except underscore ($5f or 0x5f) and the last period ($2e or 0x2e) are replaced by a percent character, followed by the two-character hexadecimal code of the escaped character. For example, the home file system name *Cañada return - 20%* becomes *Ca%96ada%20return%20%2d%2020%25*.

To generate the AppleDouble Header filename, prefix the AppleDouble Data filename with the percent character (%). Conflicts between AppleDouble Header files and AppleSingle or data files that begin with “%,” are resolved by the file’s magic number.

Since expanding characters into “%” followed by two hexadecimal digits can result in long filenames, some foreign systems may not be able to store complete 31-character filenames. Behavior under these conditions is not defined here.

Usage

AppleDouble Header files must be locked during access to ensure data integrity.

Applications that access AppleDouble files ignore all unknown entries, yet preserve them when moving or copying files.

Directories are stored as AppleDouble files if they are created by the application.

The Real Name entry is associated with file storage. File server applications typically use the reverse mapping of the foreign filename when they present names to the client. Servers do not create or read a Real Name entry.

When renaming the home file system name, the foreign filename must be renamed according to the filename conventions described in the previous sections.

Applications should not create AppleDouble Header files for preexisting foreign directories or files unless this is necessary to store AppleDouble entries.

Appendix
Updating Version 1 AppleSingle/AppleDouble Files

Applications should create AppleSingle/AppleDouble files using the version 2 format and naming conventions described in this document. Applications should understand version 1 ($00010000 or 0x0010000) format and naming conventions but should not create new files in the version 1 format.

You can update version 1 files to version 2 files by performing the following steps:

- Overwrite the version number and filler fields in the AppleSingle or AppleDouble file header.

- Replace the File Info entry (ID=7) in the version 1 file with the File Dates Info entry (ID=8) and one of the following entry IDs: Macintosh File Info (ID=10), ProDOS File Info (ID=11), or MS-DOS File Info (ID=12).