Apple II Technical Notes

Developer Technical Support

Apple IIGS
#64: Apple IIGS Installer and Installer Scripts

Revised by: Jim Luther September 1989
Written by: Jim Luther & "Jay" Schaffer July 1989

This Technical Note describes how the Apple IIGS Installer program executes script files and documents how to write script files for it. Note that some of the information in this Note is specific to Installer V1.10.
Changes since July 1989: Changed the `sourcePrefix` and `sourcePathname` field descriptions, since `sourcePrefix` must not be empty if any `sourcePathname` fields are partial pathnames.

Introduction

The Apple IIGS Installer, a utility program that is included with Apple IIGS System Software, can be used to install System Software or applications on a given volume. "Scripts" control the Installer, and they are simply lists of files with information about where and how to install those files. The user interface of the Installer is described in the *Apple IIGS System Tools Manual*. This Note describes how the Installer executes scripts and how to write scripts to install your applications.

Installer Setup on Disk

Setting up the Installer on your application disk is a simple procedure.

1. Copy the Installer program to your application disk.

2. Create a subdirectory (folder) named Scripts at the same directory level as the Installer program.
3. Copy your scripts into the Scripts subdirectory.

How the Installer Processes Scripts

The Installer reads script files into memory in their entirety, parses them, strips them of all comments, compacts them, then verifies them. It then checks the `scriptFlags` field to see if a Caution alert should be displayed. This facility permits the script writer to force the user to read the script's help message and make a choice to either continue with file manipulations or skip the installation altogether, which is especially useful when a script installation would be inappropriate on a certain volume.

The Installer then executes the script in two passes. The first pass determines if the update can be completed by calculating the total size of the files to be deleted from the destination volume and of

the files to be installed. If there is not sufficient room on the destination volume, the Installer determines the amount of additional space required to complete installation (number of blocks needed divided by two, plus one), reports this result to the user in terms of kilobytes, then terminates execution of the script. It is impossible to determine directory block requirements with complete accuracy. The Installer's space calculation algorithms are good, but they are not perfect.

If the first pass determines that there is sufficient room for the complete update, the Installer continues with the second pass, deleting and copying files in accordance with the instructions contained in the script flags. The Installer "blindly" unlocks locked files and folders, creates necessary subdirectories if they do not already exist, and replaces requested files without regard to version numbers or creation dates of existing files.

The user may terminate execution of any script (and of those which follow) by pressing the Open-Apple–Period (□-.) key combination. The Installer checks for key-down events between every file transfer and at the end of the first pass. If the user requests termination, the Installer warns of the possibility of leaving an unknown mix of file versions on the volume and gives the user the opportunity to continue with the installation or to terminate as requested. (See the "Error Handling" section for more details.)

Scripts are typically written with the ability to remove all of their related files from a particular volume (i.e., in case of an accidental installation); however, they do not have the ability to remove directories which contain files (even if the script installed them), and they can neither recover nor list files which were deleted during the installation process.

After processing all the instructions in a script, the Installer checks to see if additional scripts are selected, and, if they are, it executes them in the order in which they appear in the update selection window until all scripts are successfully completed. Once all selected scripts are completed, the Installer notifies the user that the installation or removal process was successful.

It is important to note several facts about script execution:

- Each script is processed from beginning to end as if it were the only script selected.
- If the execution of a script generates an error, or if the user terminates further processing of a script, the queue is cleared of any additional scripts waiting to be executed and control returns to the user.
- It is possible for the Installer to execute several scripts successfully before encountering one which cannot be executed due to insufficient space on the destination volume.
- All selected scripts use the folder that the user selects as the "Application Folder."

If a user installs or removes system files (i.e., tools, fonts, drivers, etc.) from the boot volume, it may create problems. Therefore, whenever a system level update occurs on a boot volume, the Installer disables all desk accessories and closes the Sys.Resources file. When the user quits the Installer after a system level update, it alerts the user of the need to restart the system, and the default response to this alert is to restart.

Error Handling

User Cancel Request

If the user cancels script execution any time after it has started (i.e., by pressing the Open-Apple–Period (□-.) key combination), the Installer treats it as an error condition since there is likely an unknown mix of file versions on the volume. In this case, the Installer gives the user the

opportunity to continue with the installation or to terminate as requested. A user-initiated cancel request is not acknowledged until the current file copy or delete request is complete. Terminating script execution also clears the queue of other scripts waiting for execution and returns control to the user.

Non-Recoverable Errors

Some errors are simply fatal. If a directory or file is corrupted, the media is bad, or the selected script is longer than 65,535 bytes, the Installer halts execution of the script and alerts the user that a fatal error has occurred with a Stop alert box. Clicking the OK button in this alert box clears the queue of other scripts waiting for execution and returns control to the user.

Script Errors and File Not Found Errors

When the Installer detects a script error or a File Not Found error, it reports the name of the source file and destination file it was processing with the normal error message. This additional information should help script writers find the offending `fileSpecification` field. If the error is associated with the header, no filename is reported. This condition clears the queue of other scripts waiting for execution and returns control to the user.

Volume Not Found Errors

Volume Not Found errors produce a dialog box prompting the user to insert the missing volume. If the user clicks the OK button, the Installer attempts the file access call again, but if the user clicks the Cancel button, the Installer flags it as an error condition, clears the queue of other scripts waiting for execution, and returns control to the user.

Script File Composition

A script is simply a list of instructions for the Installer, and it can specify that files be copied from a source volume to a destination volume (or directory, when applicable) or that files be removed from a destination volume. Script files are ASCII files (file type $04) containing printable ASCII characters (i.e., with the high-bit clear). The directory in which the Installer resides must contain a directory named Scripts, in which all script files visible to that copy of the Installer must be located. Script files may not exceed 65,535 bytes in length. Any attempt to execute a script larger than this size produces a non-recoverable error.

A script consists of a `header` field followed by any number of `fileSpecification` and `comment` fields. These fields are separated by tildes (~). Two consecutive tildes signal the end of the script, and any additional characters past the end of script marker are ignored. Figure 1 shows the syntax diagram for a script.

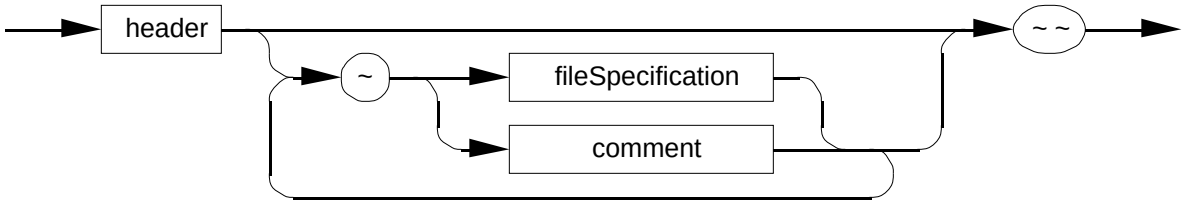

Figure 1–Script Syntax Diagram

header Field

The `header` field consists of the `scriptIdentifier`, `scriptVersion`, `scriptFlag`, `scriptName`, and `scriptHelp` fields, and it may also contain an optional `sourcePrefix` field. These fields supply the installer with general information about the script file. No comments are permitted within the `header` field. Figure 2 shows the syntax diagram for the `header` field.

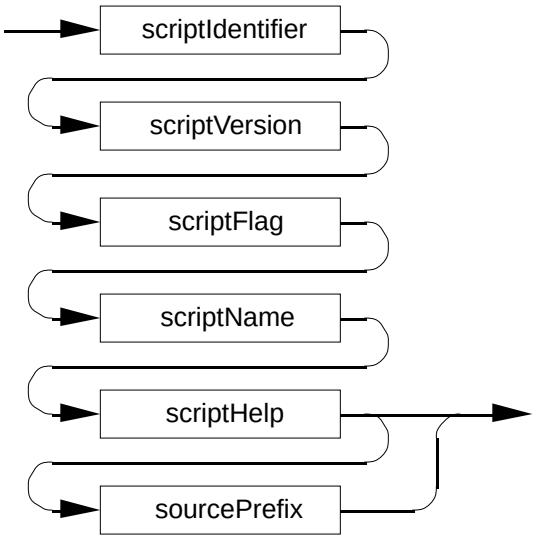

Figure 2–header Field Syntax Diagram

The `scriptIdentifier` field identifies the text file as a script file, and it consists of eight characters (“SCRIPT” followed by two carriage returns, or 53 43 52 49 50 54 0D 0D in hexadecimal). Figure 3 shows the syntax diagram for the `scriptIdentifier` field.

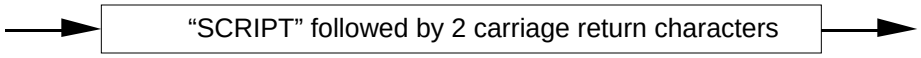

Figure 3–scriptIdentifier Field Syntax Diagram

The `scriptVersion` field defines the minimum version of the Installer program that can read and execute the instructions in this script file. It should normally consist of seven characters (“V1.10” followed by two carriage returns, or 56 31 2E 31 30 0D 0D in hexadecimal).

Version 1.0 of the Installer moves **only** the data fork and does not return an error. For compatibility with the original release of the Installer, the value of `scriptVersion` is V1.00. Scripts which move extended files (i.e., files with resource forks) or work with an AppleShare volume **must** have a `scriptVersion` of V1.10. Figure 4 shows the syntax diagram for the `scriptVersion` field.

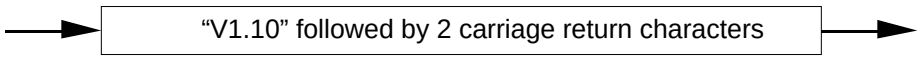

Figure 4–scriptVersion Field Syntax Diagram

The `scriptFlag` field defines the directory requirements of the script file. The first character of the `scriptFlag` field must be either the uppercase character “R” (indicating that the installation must occur at the root directory, such as in a System Software update) or the uppercase character “X” (indicating that the user must specify the directory where installation should take place).

The second character of the `scriptFlag` field must be either an uppercase or lowercase character “R” (indicating that the Remove command is valid for this script) or an uppercase or lowercase character “N” (indicating that the Remove command is not valid and the button should be dimmed and inactive). If this character is lowercase, before any file manipulations begin, the Installer displays a Caution alert with the contents of the `scriptHelp` field and button controls to permit the user to choose whether to execute the script or to skip it and go to the next script, if any.

For example, a `scriptFlag` field might contain the following four characters: “Rr” followed by two carriage returns, or 52 52 0D 0D in hexadecimal. Figure 5 shows the syntax diagram for the `scriptFlag` field.

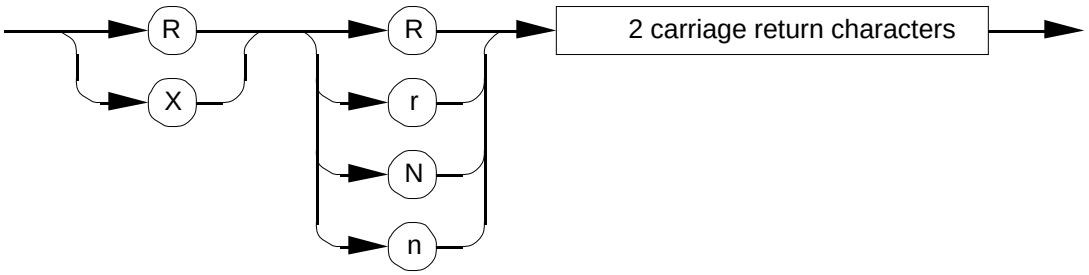

Figure 5–scriptFlag Field Syntax Diagram

The `scriptName` field defines the name of the script as it appears in the Installer’s script selection window. It is recommended that care be taken to use a name that fits within the display window. This field consists of any number of characters ending with a carriage return and may not include a tilde or carriage return. An example of `scriptName` might be: “Example Script” followed by a carriage return, or 45 78 61 6D 70 6C 65 20 53 63 72 69 70 74 0D in hexadecimal. Figure 6 shows the syntax diagram for the `scriptName` field.

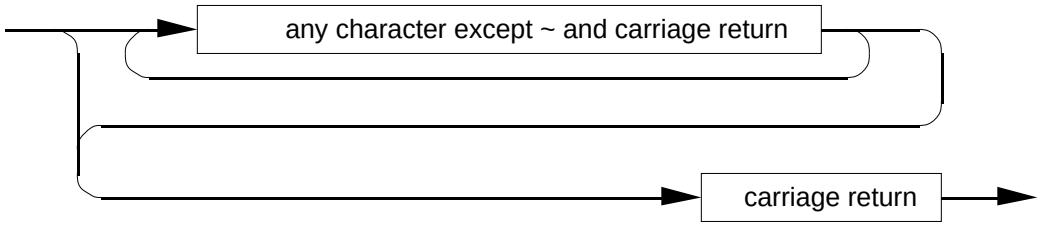

Figure 6–scriptName Field Syntax Diagram

The `scriptHelp` field defines the text which appears when the user clicks the Help button. It is recommended that care be taken to ensure the text fits within the help window. This field consists of any number of characters ending with two backslashes (\\) and a carriage return. It may not include two consecutive backslashes or a tilde; however, it may include carriage returns. An example of `scriptHelp` might be: "Help\\" followed by a carriage return, or 48 65 6C 70 5C 5C 0D in hexadecimal. Figure 7 shows the syntax diagram for the `scriptHelp` field.

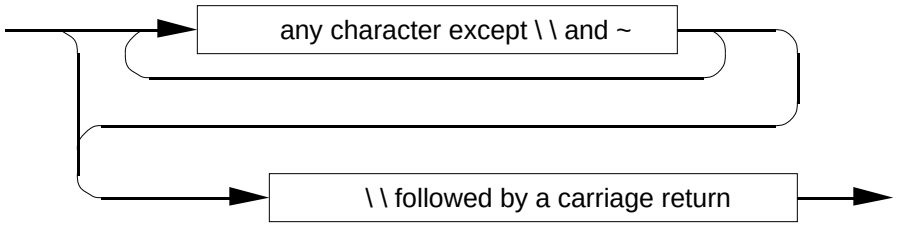

Figure 7–scriptHelp Field Syntax Diagram

The optional `sourcePrefix` field is the prefix used with source files defined by partial pathnames. Either slashes (/) or colons (:) may be used as the pathname separator character. If there is no `sourcePrefix`, this entry must be empty. If no `sourcePrefix` is specified, all `sourcePathname` fields used within `fileSpecification` fields must be full pathnames. An example of a `sourcePrefix` might be: ":System.Disk:System", or 3A 53 79 73 74 65 6D 2E 44 69 73 6B 3A 53 79 73 74 65 6D in hexadecimal. Figure 8 shows the syntax diagram for the `sourcePrefix` field. *GS/OS Reference* defines legal pathnames and prefixes.

Figure 8–sourcePrefix Field Syntax Diagram

fileSpecification Field

A `fileSpecification` field contains the instructions to copy a file to or remove a file from the destination volume (or directory, when applicable). A `fileSpecification` field is composed of the `fileSpecWorkspace`, `fileSpecFlags`, `fileTypeAuxType`, `createDateTime`, `sourcePathname`, and `destinationPathname` fields. The script may contain as many `fileSpecification` fields as necessary. Figure 9 shows the syntax diagram for the `fileSpecification` field.

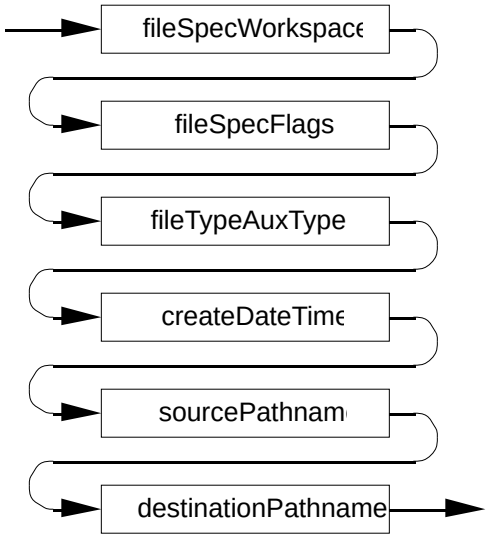

Figure 9–fileSpecification Field Syntax Diagram

The `fileSpecWorkspace` field is 16 bytes that the Installer uses for work space, it can contain any character except a tilde, and it may not begin with a tilde or an asterisk (*). It is suggested that 15 readable characters followed by a carriage return might be easiest to see and count. An example of `fileSpecWorkspace` might be: ":::Workspace:::" followed by a carriage return, or 3A 3A 3A 57 6F 72 6B 73 70 61 63 65 3A 3A 3A 0D in hexadecimal. Figure 10 shows the syntax diagram for the `fileSpecWorkspace` field.

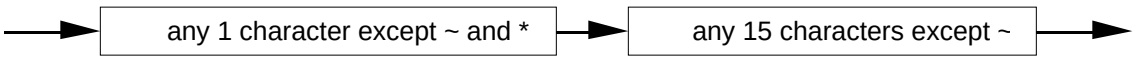

Figure 10–fileSpecWorkspace Field Syntax Diagram

The `fileSpecFlags` tell the Installer what this `fileSpecification` does. The `fileSpecFlags` field consists of the `requiredFlag` field followed by the `optionalFlags` field and a carriage return. Figure 11 shows the syntax diagram for the `fileSpecFlags` field.

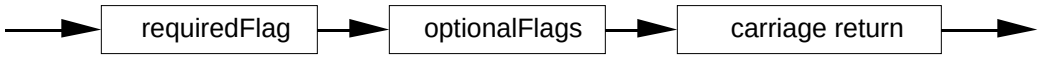

Figure 11–fileSpecFlags Field Syntax Diagram

The `requiredFlag` field tells the Installer what to do with this `fileSpecification` when the Install or Remove buttons are used. The `requiredFlag` field must start with only one of the following characters: 1, 2, 3, or 4, and it must end with a carriage return. Any number of characters (except tilde and carriage return) may fall between the flag character and the ending

carriage return. These additional characters are ignored by the Installer, making it possible to place comments within a `requiredFlag` field. Figure 12 shows the syntax diagram for the `requiredFlag` field.

The four `requiredFlag` characters tell the installer the following:

1 If the user clicks the Install button, delete the `destinationPathname` from the destination volume, if it exists, and copy the file from the source volume. If the user clicks the Remove button, delete the `destinationPathname` from the destination volume, if it exists.
2 If the user clicks the Install button, delete the `destinationPathname` from the destination volume, if it exists, and copy the file from the source volume. If the user clicks the Remove button, do nothing.
3 If the user clicks the Install button, delete the `destinationPathname` from the destination volume, if it exists. If the user clicks the Remove button, delete the `destinationPathname` from the destination volume, if it exists.
4 If the user clicks the Install button, delete the `destinationPathname` from the destination volume, if it exists. If the user clicks the Remove button, do nothing.

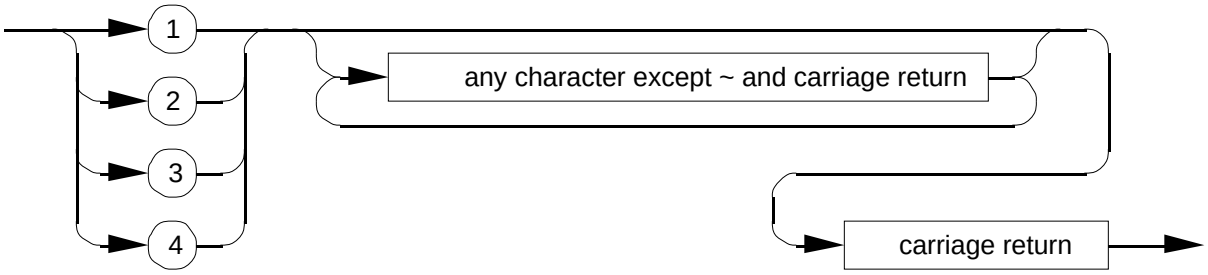

Figure 12–requiredFlag Field Syntax Diagram

The `optionalFlags` field gives the Installer additional duties to perform with this `fileSpecification` when the Install or Remove buttons are used. The five option fields, B, C, D, F, and U (must be uppercase), within the `optionalFlags` field are formatted the same as the `requiredFlag` field. Figure 13 shows the syntax diagram for the `optionalFlags` field.

The five `optionalFlags` characters tell the installer the following:

B This flag instructs the Installer to replace the boot code on blocks zero and one of the destination volume. The boot code replacement `fileSpecification` is reserved for use by Apple Computer, Inc.
C The creation date and time of the file designated by the `sourcePathname` field must match the `createDateTime` entry in this `fileSpecification` field.

D The designated `destinationPathname` should be deleted if, and only if, it has a creation date and time that is older than `createDateTime`. This flag must be used with a "4" `requiredFlag`.

F The file type and auxiliary type of the file designated by the `sourcePathname` must match the `fileTypeAuxType` field in this `fileSpecification` field.

U Update (replace) the existing `destinationPathname` only if it exists. This flag must be used with a "1" or a "2" `requiredFlag`.

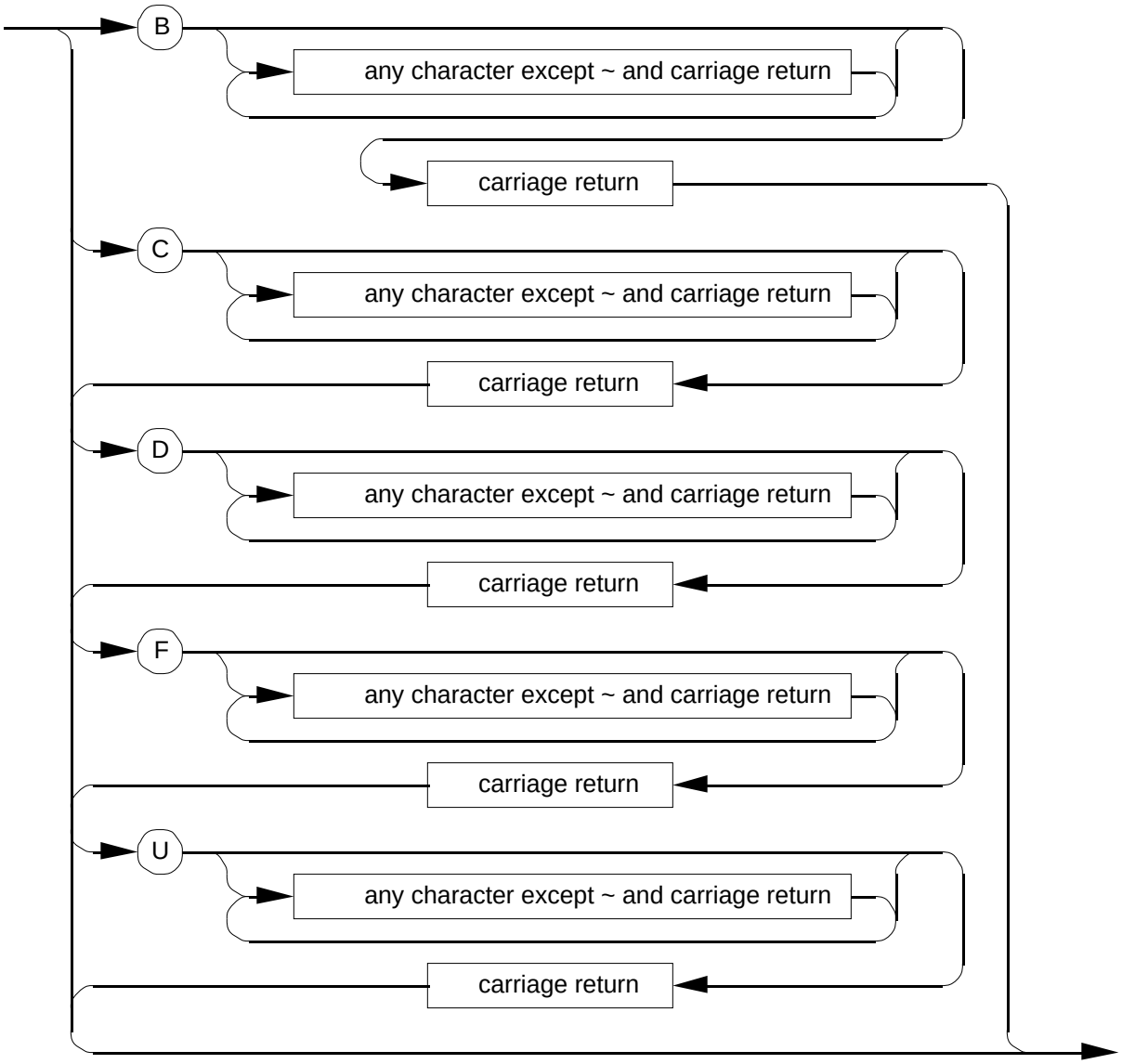

Figure 13–optionalFlags Field Syntax Diagram

The `fileTypeAuxType` field is used if the "F" `optionalFlags` field is present in the `fileSpecification` field. If the `fileTypeAuxType` field is used, it must start with a `fileType` field and an `auxType` field and must end with a carriage return. Any number of characters (except tilde and carriage return) may fall between the `auxType` field and the ending carriage return. These additional characters are ignored by the Installer, making it possible to place comments within the `fileTypeAuxType` field. If the "F" `optionalFlags` field is not used, then the fileTypeAuxType field must be only a carriage return. For a list of current file types and auxiliary types, see the Apple II File Type Notes. Figure 14 shows the syntax diagram for the `fileTypeAuxType` field.

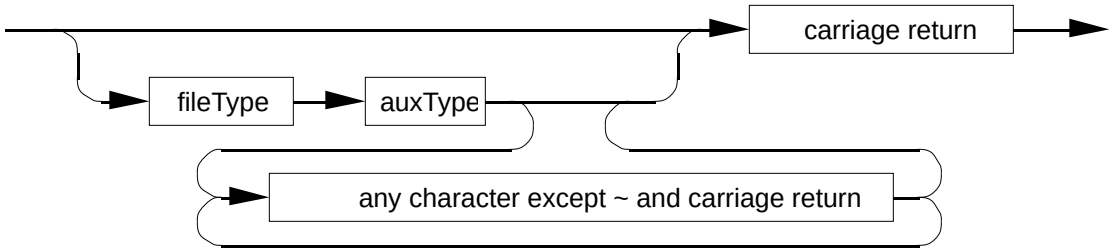

Figure 14–fileTypeAuxType Field Syntax Diagram

The `fileType` part of the `fileTypeAuxType` field consists of four, and only four, hexadecimal digits. These four digits identify a GS/OS file type if the "F" `optionalFlags` field is present in the `fileSpecification` field. An example of `fileType` might be: "00B3", or 30 30 42 33 in hexadecimal. Figure 15 shows the syntax diagram for the `fileType` field.

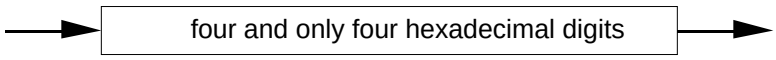

Figure 15–fileType Field Syntax Diagram

The `auxType` part of the `fileTypeAuxType` field consists of eight, and only eight, hexadecimal digits. These eight hexadecimal digits identify a GS/OS auxiliary type if the "F" `optionalFlags` field is present in the `fileSpecification` field. An example of `auxType` might be: "00000000", or 30 30 30 30 30 30 30 30 in hexadecimal. Figure 16 shows the syntax diagram for the `auxType` field.

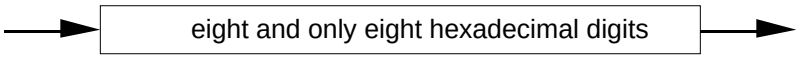

Figure 16–auxType Field Syntax Diagram

The `createDateTime` field is used if the "C" or "D" `optionalFlags` fields are present in the `fileSpecification` field. If the `createDateTime` field is used, it must start with a `date` field, a single space and a `time` field and must end with a carriage return. Any number of characters (except tilde and carriage return) may fall between the `time` field and the ending carriage return. These additional characters are ignored by the Installer, making it possible to place comments within the `createDateTime` field. If the "C" or "D" `optionalFlags` fields are not used, then the `createDateTime` field must be only a carriage return. Figure 17 shows the syntax diagram for the `createDateTime` field.

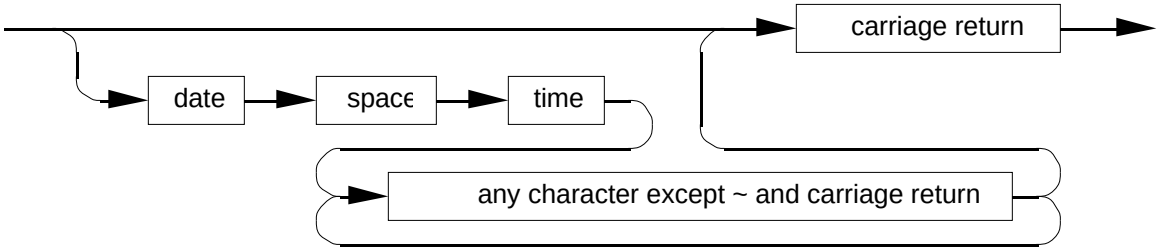

Figure 17–createDateTime Field Syntax Diagram

The `date` subfield of the `createDateTime` field is nine ASCII characters consisting of the day of the month, a space, a three-character month abbreviation, a space, and the year. The day of the month is a two-character number between 01 and 31. The month abbreviation may be "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", or "Dec" in any combination of uppercase and lowercase characters. The year is a two-character number between 00 and 99. An example of the `date` subfield might be: "31 Mar 57", or 33 31 20 4D 61 72 20 35 37 in hexadecimal. Figure 18 shows the syntax diagram for the `date` subfield.

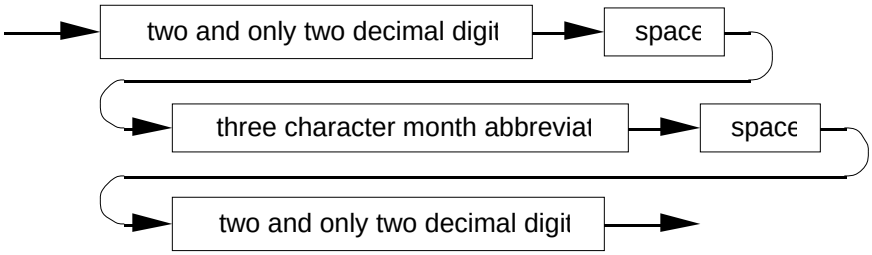

Figure 18–date Field Syntax Diagram

The `time` subfield of the `createDateTime` field is five ASCII characters consisting of the military format hour of the day, a colon, and the minute of the hour. The hour of the day is a two-character number between 00 and 23. The minute of the hour is a two-character number between 00 and 59. An example of the `time` subfield might be: "08:30", or 30 38 3A 33 30 in hexadecimal. Figure 19 shows the syntax diagram for the `time` subfield.

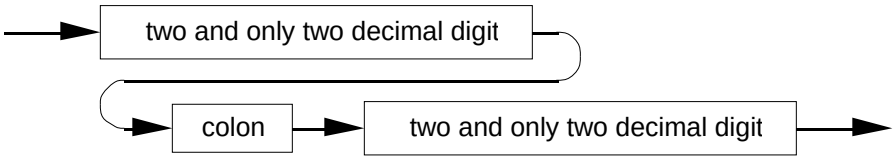

Figure 19–time Field Syntax Diagram

The `sourcePathname` field describes the name and location of the source file. The `sourcePathname` field consists of a valid GS/OS pathname followed by a carriage return. If the `sourcePathname` is a partial pathname, the `sourcePrefix` in the `header` field is used to complete the full pathname. If no `sourcePrefix` is specified in the `header` field, all `sourcePathname` fields must be full pathnames. If the `fileSpecFlags` indicate removal only, then the `sourcePathname` is a carriage return only. No optional comments are permitted in this field. Figure 20 shows the syntax diagram for the `sourcePathname` field. *GS/OS Reference* defines legal pathnames and prefixes.

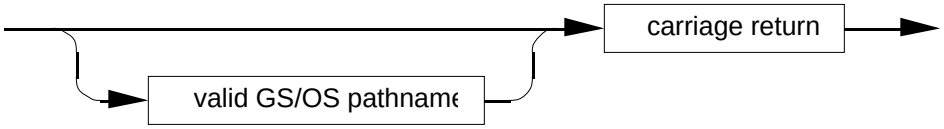

Figure 20–sourcePathname Field Syntax Diagram

The `destinationPathname` field describes the name and location of the destination file. The `destinationPathname` field consists of a valid GS/OS partial pathname (the prefix has already been set by the Installer to the location of the destination directory, either the root directory or a user selected directory) followed by a carriage return. No optional comments are permitted in this field. Figure 21 shows the syntax diagram for the `destinationPathname` field. *GS/OS Reference* defines legal pathnames and prefixes.

Note that GS/OS now allows filenames to contain both uppercase and lowercase characters. Although filenames are not case sensitive, you should be consistent in your use of uppercase and lowercase usage in the `destinationPathname` field. Whatever you use here is what everyone sees.

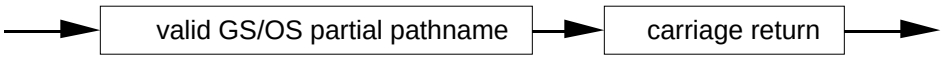

Figure 21–destinationPathname Field Syntax Diagram

comment Field

The `comment` field allows commenting script files. A `comment` field must begin with an asterisk. The Installer ignores all characters within a `comment` field, except tilde, and the `comment` field ends at the first tilde encountered. Figure 22 shows the syntax diagram for the `comment` field.

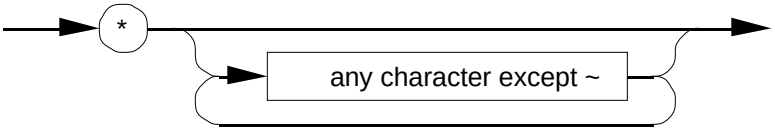

Figure 22–comment Field Syntax Diagram

Examples

Now that the script language is described, it's time to look at a couple of example scripts. The first example, CD-ROM from the System.Tools disk, installs the files necessary for you to use CD-ROM drives. The CD-ROM script is an example of using the Installer to install or update existing software. The second example, Advanced Disk Utility from the System.Tools disk, installs the files necessary to update the Advanced Disk Utility program. The Advanced Disk Utility script is an example of using the Installer to install an application in any directory on the destination volume. In both examples (Examples 1 and 2), carriage returns are shown with a paragraph mark (¶) since they are used as delimiters within scripts.

The CD-ROM Script

The `header` field starts with "SCRIPT" to identify this text file as a script file. The `scriptVersion` is "V1.10" because this script may have to copy the resource fork of a file. The `scriptFlag` field is "RR", which tells the Installer to install at the root directory level and that the Remove button is valid for this script. The second "R" character in the `scriptFlag` field is uppercase, which tells the Installer **not** to display a Caution alert with the contents of the `scriptHelp` field. The `scriptName` field is "CD-ROM". The `scriptName` is shown in the Installer's list of scripts. The `scriptHelp` field (everything between the `scriptName` field and the "\\" delimiter) is the text that will be displayed if the Installer's Help button is used. The `sourcePrefix` is ":SYSTEM.TOOLS". That is the name of the volume where the source files for this update are found.

After the `header` field, there is a single `comment` field and then five `fileSpecification` fields. The `comment` field starts at the asterisk after the first tilde and ends at the next tilde. All five `fileSpecification` fields start with the suggested 16-byte `fileSpecWorkSpace` (":::WorkSpace:::¶") and end at the next tilde.

The first, fourth, and fifth `fileSpecification` fields use the "1" `requiredFlag`. This flag tells the Installer to copy the `sourcePathname` to the `destinationPathname` if the Install button is used, or to delete the `destinationPathname` if the Remove button is used. Notice the three blank lines after the "1" `requiredFlag`. The first blank line marks the end of the `fileSpecFlags`. The `fileTypeAuxType` field, the second blank line, is blank because the "F" `optionalFlags` field is not used. The `createDateTime` field, the third blank line, is blank because the "C" and "D" `optionalFlags` are not used.

The second `fileSpecification` field uses the "3" `requiredFlag` to tell the Installer to delete the `destinationPathname`, "System:Drivers:SCSI.Driver", if either the Install or the Delete button is used. SCSI.Driver is the interim SCSI driver from System Software 4.0. The `sourcePathname` field, the fourth blank line after the "3" `requiredFlag`, is not

needed since the “3” `requiredFlag` is used.

The third `fileSpecification` field uses the “2” `requiredFlag` to tell the Installer to delete the `destinationPathname`, “System:Drivers:SCSI.Manager” if the Install button is used. The Installer does **not** delete the `destinationPathname` if the Remove button is used. The “2” `requiredFlag` prevents this script from removing SCSI.Manager, which might have been installed by another script.

Two consecutive tildes after the fifth `fileSpecification` field signal the end of this script.

```
SCRIPT¶
¶
V1.10¶
¶
RR¶
¶
CD-ROM¶
This script installs the files necessary for you to use CD-ROM drives.  The selected disk must be a
startup disk.\\¶
:SYSTEM.TOOLS~*¶
This is the Installer script necessary to move the CD-ROM files from :SYSTEM.TOOLS to the user's
startup disk.¶
~:::Workspace:::¶
1¶
¶
¶
¶
System:FSTs:HS.FST¶
System:FSTs:HS.FST¶
~:::Workspace:::¶
3¶
¶
¶
¶
¶
System:Drivers:SCSI.Driver¶
~:::Workspace:::¶
2¶
¶
¶
¶
System:Drivers:SCSI.Manager¶
System:Drivers:SCSI.Manager¶
~:::Workspace:::¶
1¶
¶
¶
¶
System:Drivers:SCSICD.Driver¶
System:Drivers:SCSICD.Driver¶
~:::Workspace:::¶
1¶
¶
¶
¶
System:Desk.Accs:CDRemote¶
System:Desk.Accs:CDRemote¶
~~
```

Example 1–CD-ROM Script

The Advanced Disk Utility Script

The `header` field starts with "SCRIPT" to identify this text file as a script file. The `scriptVersion` is "V1.10" because this script may have to copy the resource fork of a file. The `scriptFlag` field is "XR", which tells the Installer the user must specify the directory where the installation should take place and that the Remove button is valid for this script. The second character (R) in the `scriptFlag` field is uppercase, which tells the Installer **not** to display a Caution alert with the contents of the `scriptHelp` field. The `scriptName` field is "Advanced Disk Utility". The `scriptName` will be shown in the Installer's list of scripts. The `scriptHelp` field (everything between the `scriptName` field and the "\\" delimiter) is the text that will be displayed if the Installer's Help button is used. The `sourcePrefix` is ":SYSTEM.TOOLS". That is the name of the volume where the source files for this update are found.

After the `header` field, there is a single `comment` field then one `fileSpecification` field. The `comment` field starts at the asterisk after the first tilde and ends at the next tilde. The `fileSpecification` field starts with the suggested 16-byte `fileSpecWorkSpace` (":::WorkSpace:::¶") and ends at the next tilde.

The `fileSpecification` field uses the "1" `requiredFlag`. This tells the Installer to copy the `sourcePathname` to the `destinationPathname` if the Install button is used or to delete the `destinationPathname` if the Remove button is used.

Two consecutive tildes signal the end of this script.

```
SCRIPT¶
¶
V1.10¶
¶
XR¶
¶
Advanced Disk Utility¶
This script installs the files necessary to update the Advanced Disk Utility program.  These files
will be installed on the selected disk.\\¶
:SYSTEM.TOOLS~*¶
This is the Installer script necessary to update the Advanced Disk Utility file from :SYSTEM.TOOLS
to the user's disk.¶
~:::Workspace:::¶
1¶
¶
¶
¶
Adv.Disk.Util¶
Adv.Disk.Util¶
~~
```

Example 2–Advanced Disk Utility Script

Further Reference

- *Apple IIGS System Tools Manual*
- *GS/OS Reference*