Technical Note PT520
Installer Q&As

<table>
<tr><td>

CONTENTS

Introduction

References

Downloadables

</td><td>

This Technical Note contains a collection of archived Q&As relating to a specific topic--questions sent the Developer Support Center (DSC) along with answers from the DSC engineers. Current Q&A's can be found on the Macintosh Technical Q&A's web site.

</td></tr>
</table>

[Oct 01 1990]

Solutions for aborted network install

My installation process works great from a floppy disk; however, when I try to create a network install from these disks, the Installer reports that one of the file can't be found and aborts the installation. What's up?

The following conditions probably are true for your installation:

1. The AppleShare volume being used for the network install is a Novell server.
2. Within the Installer script source file for some `'infs'` source file specification resource, the creation date/time stamp field has been set to 1 or some other nonzero odd value (1 tells `ScriptCheck` to fill in the field for you).
3. The creation date/time stamp for a file specified in condition 2 above falls on an odd second.

So what do these three conditions have to do with each other? Condition 2, the nonzero value for the date/time stamp field, tells the Installer to verify the files actual date/time with that stored in the `'infs'` resource. If the values don't match, the Installer reports that the file couldn't be found. On Novell servers, the time stamp granularity is two seconds. When you copy a Macintosh file whose time stamp ends in an odd second to a Novell server, the external file system for the Novell server adjusts the time stamp to an even second. This one second difference is enough to tell the Installer that the found file doesn't match. This assumes that the server and the workstation have their system clocks set to within 15 minutes of each other. The `'incd'` resource Q&A in the Installer Appendix describes how the time difference is handled when the two clocks differ by more than 15 seconds.

There are a couple of solutions to this problem. The first is to set the value of the date/time stamp fields for all `'infs'` resources to 0. This means that the Installer will not verify that the right file is being used, except by the name and location of a source file. Another solution is to use the MPW tools or ResEdit to make sure that the date/time stamp for all source files is even. The Installer Q&A Appendix discusses another special case to consider.

Why Finder requests program disks after installation

Date Written: 2/5/93

Last reviewed: 7/2/93

I just wrote an installer script for the Apple Installer version 3.4. After the installation is complete and I click the Quit button, the Finder insists on mounting both of the disks that were used in the installation. It gives me the annoying dialog "Please insert the disk Install 1" and "Please insert the disk Program 1." Before I did the installation, only the Install 1 disk was mounted by the Finder. Why is this happening?

This situation typically occurs when the installer disk sets were created under System 7 and the installation is happening under system software version 6.0.x. Its cause is that the system is trying to update the second disk's desktop. The solution in this case is to insert the disk into a system running version 6.0.x before performing the installation.

Another possible cause for this problem is that the second disk opens to display a window by default. When the Installer quits, there's still "residue" Finder information to be written to the second disk, which has since been ejected. The solution in this case is to insert the second disk with the write tab lock disabled, and close any open window associated with that disk.

Order of actAfter and Installer 'inaa' and 'inat' atoms

Date Written: 1/5/93

Last reviewed: 4/1/93

When an installation package contains

```
'inaa', 205,/*Restore 'eadr' resource*/
```

is the order of occurrence of these two atoms important? Will the `'inaa'` always happen *after* or *before* the `'inat'`? The documentation isn't clear whether the `actAfter` is really after *everything;* my experience shows it isn't.

You're right, the `'inat'` resources are installed following the call to the last postinstallation action atom.

No difference between addAuditRec and installing an 'inat' resource

Date Written: 1/5/93

Last reviewed: 6/14/93

What's the difference between an `addAuditRec` and installing an `'inat'`? The documentation doesn't mention the two in the same sentence.

There is no difference. The `AddAuditRec` clause and the `'inat'` resource do exactly the same thing.

Installer sample shows how to install fonts by strikes

Date Written: 11/17/92

Last reviewed: 6/14/93

Is there an Installer sample demonstrating how to install fonts by strikes?

Yes, check the Installer samples folder on the latest *Developer CD Series* disc. By the way, if you ever get the error "There's a problem with the Installer Document <document name>" and you're installing fonts, one possible reason for the error is that the script doesn't have the sizes of the split resource size matching those of the actual resource exactly.

Installing to the startup System Folder

Date Written: 8/17/92

Last reviewed: 6/14/93

When installing on a system that has multiple volumes and a user selects a nonboot volume on which to perform the installation, is there a way to specify that specific files or resources get installed to the boot blessed folder and system?

Unfortunately, no. The Installer is designed to install all components of the installation to the target volume specified by the user. The Installer script writer can specify that a blessed system folder exist on the target volume, or that specific files exist before installing a resource; however, these actions work only for a single volume. This suggestion has been passed to the Installer engineering team for their consideration for a future version of the Installer.

A sample action atom code resource is included with the Installer kit on the latest Developer CD to demonstrate a method for installing files to the boot blessed system folder. The code resource takes as an argument, the resource ID of an 'infa' file atom to access the source and target `'infs'` file specification resources. There are several caveats associated with using this sample; however, it may provide a starting sample for implementing such functionality.

Implementing Installer file compression

Date Written: 8/17/92

Last reviewed: 6/14/93

Does the Installer provide any utility routines for compressing files, and decompressing them on installation?

The Installer does not provide any built-in capabilities for compressing files nor for decompressing files on installation. The Installer does implement an action atom mechanism, by which the developer can write custom code resources to run before or after an installation process. Based on this mechanism, several developers have implemented custom code resources to handle the decompression of compressed files. A typical process is to compress the file using a compression application. The Installer script is written to install the compressed files to the target disk. The decompression code resource is run after the installation process is complete to decompress the files. A list of developers who license such decompression code resources is available on request from Developer Support. Please note that the list does not constitute any endorsement by Apple of these products.

If you've already implemented a file decompression action atom code resource, you'll find the new MPW `ScriptCheck` tool version 3.4b2 includes some important feature enhancements. The most significant feature is that `ScriptCheck` version 3.4 can set to leave the size field of the 'infa', file atom resource alone, if the size is nonzero. The Installer uses the size field to determine the disk space required for the installation. An important problem experienced by script writers concerned the fact that `ScriptCheck` overrode the size field setting of the 'infa' resource to that of the size of the compressed file. If left alone, the Installer would preflight the target disk for simply the amount of disk space to copy the compressed file only. This forced script writers to edit the script afterwards to replace the size field with the actual value of the uncompressed file.

Another important enhancement of `ScriptCheck` 3.4b2 is that it now recognizes the 'inff' font family atom, and the version 1 action atom resource. More information about `ScriptCheck` 3.4b2 is in the `ScriptCheck` Guide for 3.4b2 now available on the latest Developer CD. Note that `ScriptCheck` 3.4b2 is compatible with Installer 3.3 and 3.4 scripts--that is, scripts whose file types and creators are 'bbkr'.

Opening a Macintosh Finder window from an Installer action atom

Date Written: 8/14/92

Last reviewed: 11/30/92

My installer creates a folder on a user's hard disk and copies the necessary files into it. My final action atom moves the folder onto the desktop and sets its size and location. I'd also like to be able to open the folder. I call `PBGetCatInfoSync` to get the data into a `PBGetCatInfoSync` record. Where is the state of a folder (open/closed) stored, and can I set one of the parameters in the `PBGetCatInfoSync` and then call `PBSetCatInfoSync` to solidify the change? Using the installer to copy an open folder to the user's drive is unacceptable because of the size and nature of the program I'm installing.

There's no solution for System 6; the Finder data structures are private, and there's no call to open a folder. In System 7, you can send the Finder an Open Selection Apple event. This is described in a HyperCard stack called FinderEvents on the *Developer CD Series* disc. The stack also contains the source code for the XCMD used to demonstrate the Finder events. There's another sample that you should see as well: SendFinderOpen in the Snippets folder.

Installer cleanUpCancel mechanism

Date Written: 7/20/92

Last reviewed: 11/1/92

The document for Installer 3.2 describes the use of the `whichStage` field (of the action atom parameter block) to determine what stage the Installer is in (before, after, or `cleanUpCancel`). I want to use the `cleanUpCancel` stage to do some cleaning up of a preinstallation action atom should the installation be cancelled. If I cancel (at any point during) the installation, my action atom is not run, however, only the `cleanUpCancel` code executed. Determining what stage your action atom is running in seems straightforward, but how does the Installer differentiate between "running" and "sending a cancel message" to an action atom? Are there 'inaa' resource flags that need to be set to enable an action atom to run during cleanup?

It turns out that this feature was broken under Installer 3.2. Under Installer 3.3, the mechanism works; however, the way it works is not clearly documented. One might think that the `cleanUpCancel` mechanism applies only to the code resource in which it's implemented, that you would need to install a procedure to act on cleanUpCancel in each action atom that gets installed. In fact, the mechanism is independent of whether an action atom is successful or not under Installer 3.3

If, for whatever reason, an installation is aborted, the Installer does its own cleanup, then searches for all action atoms marked by the flag actAfter. These action atoms are then sent the `cleanUpCancel` message. In fact, there need only be one action atom to respond in such a manner. However, to be called, the `'inaa'` resource that calls the code resource *must* have the "actAfter" flag set.

To resolve your situation, there are two things to be done: Use Installer 3.3, *and* include a second `'inaa'` resource marked as "actAfter" and have it reference the same code resource as the original `'inaa'`. Modify the code resource so that it checks the `whichStage` field and executes the cleanup code only if the `whichStage` field is set to `cleanUpCancel.` In fact a symptom that is reported on occasion is the fact that postinstallation action atom code resource sometimes gets executed twice. This symptom occurs because an error was detected sometime during or after the execution of the postinstallation action atom and the code resource does not distinguish between stages. If, for instance, the postinstall code resource reported an error, the Installer would then find that the action atom is set for `actAfter` and call the code resource a second time with the whichStage field set to `cleanUpCancel.`

Use preinstallation action atom to protect user prefs file

Date Written: 2/25/92

Last reviewed: 6/14/93

How can I keep the Macintosh Installer script from replacing a newer file? The various flag settings that I've tried always result in the target file being replaced by the source file.

The Installer uses the creation date/time stamp (as opposed to the modification date/time stamp) to determine whether the target file is newer than the source file. If the target file has the same or earlier date/time stamp as does the source file on the installation disk, the file is replaced.

To control the installation of a file based on modification date requires an action atom. The function of the action atom code resource could range from halting the installation if the target file is greater than the source file, to performing the actual file copy. Unfortunately, there isn't a method for simply bypassing the installation of a single file or package. The Installer engineering group is looking into ways of handling this feature for a future release of the Installer.

Macintosh Installer 3.2 & 3.3 live installation handling

Date Written: 3/2/92

Last reviewed: 8/19/92

I'm writing a preinstallation action atom to save some information from an existing Macintosh file that my script will later replace. When I do a live install, the file appears to have already been deleted. What's happening?

Installers 3.2 and 3.3 treat live installations differently from installations to nonboot volumes. To be able to restore the boot volume, should the installation be canceled or be unsuccessful, the Installer moves all files designated for replacement, deletion, or modification into the Installer Temp folder in the System Folder. The preinstallation action atom can make use of two parameters passed in the parameter block to access files in the temporary folder. The `didLiveUpdate` flag indicates whether a live installation has been selected. The `installerTempDirID` is the `DirID` of the temporary folder. Your action atom code resource should check the `didLiveUpdate` flag. If true, then search for the desired file in the desired folder specified by `installerTempDirID.`

Bypassing Macintosh Installer's creation date check

Date Written: 1/23/92

Last reviewed: 6/14/93

How can I get the Macintosh Installer to bypass checking the creation date of source files when preflighting an installation?

Normally, the Installer checks that the source files exist by the name specified in the `'infs'` resource and that the creation date/time stamp matches that specified in the resource as well. To bypass the second check, set the creation date field to zero. In addition, do *not* use the -d switch with `ScriptCheck,` which forces updating of the date field.

Macintosh resources: Dynamic installation and deletion

Date Written: 1/23/92

Last reviewed: 8/19/92

I've written an Installer script that has an `'infa'` file atom to copy a new file, and then uses `'inra'` resource atoms to delete resources from that file. When I finish the installation process and open the file with ResEdit, the file still contains the resources designated for deletion. Why?

When performing an installation, the Installer performs these general steps:

1. The Installer identifies all the resource and file atoms to be installed and determines whether there is sufficient target disk space. The Installer also takes into consideration that files designated for deletion will be copied into the temporary folder (for live installations only), so that the installation can be undone.
2. Preinstallation action atoms are run.
3. The Installer deletes all specified resources and files, (or moves them into the temporary folder for live installs), plus asks the user whether newer files will be replaced. After performing this function, the Installer

assumes all deletions are finished.
4. The Installer performs the installation of all designated files and resources.

The Installer handles the resource deletions first. Since the file doesn't exist or is already deleted, the deletion resource atoms are considered successful. The Installer assumes the installed files are already set as needed; that there's no need to delete resources from the file. As a result, resources, specified for deletion on a newly installed file, aren't deleted.

How Installer computes target volume storage size requirement

Date Written: 12/19/91

Last reviewed: 8/19/92

How does the Macintosh Installer compute the storage size requirement for a target volume?

The Installer creates a list of all files and resources that are to be installed. It then goes to each corresponding `'inra'` and `'infa'` resource (and 'inff' resource for version 3.3), and sums up the size fields. If a live install is being performed, a check is also made to determine the size of active files, such as the System file, that need to be duplicated. The sizes of such files are added to the installation size requirement. For this reason, it's difficult to perform a live install onto a floppy startup disk, even if for a small resource. There is generally insufficient floppy disk space to make a copy of the active System file.

Macintosh Installer and Apple event support

Date Written: 12/10/91

Last reviewed: 6/14/93

Does the Macintosh Installer support Apple events? We would like to launch Installer and have it install automatically when we send it the appropriate Apple event.

Installer Version 3.4 and earlier does not support Apple events; however, support for Apple events is being considered for a future version of the Installer.

Files for Installer postinstallation action atom

Date Written: 12/10/91

Last reviewed: 6/14/93

When a postinstallation action atom gets control, can we assume that all of the files the atom will work with are on the hard disk?

You may assume that all of the file and resource atoms included in the package resources have been fired and all files and resources designated for installation by these atoms have been copied to the target disk.

Installer and finding a source file by type and creator

Date Written: 11/27/91

Last reviewed: 6/14/93

Can a Macintosh Installer script be written so a source file is found by its file and creator type rather than by name?

No. The Installer is not designed to find a source file by type and creator. However, we've passed your suggestion along to the the Installer engineering team.

checkFileVersion clause and 'inrl' resource

Date Written: 9/30/91

Last reviewed: 6/14/93

How can I check for the existence of a minimum Macintosh file version?

Use the `checkFileVersion` clause as part of the `'inrl'` Rules Framework resource. The format of the minimal-version parameter is shown in the InstallerTypes.r file as "#define Version." The most common difficulties are in remembering that BCD values are required, and how to deal with two-digit version numbers. Some samples follow.

Assuming that the `'infs'` target-filespec resource for the System file is 1000, then use the following clause to check for System 6.0.5:

Assuming that the `'infs'` target-filespec resource for the Finder file is 1001, then use the following clause to check for Finder 6.1.5:

Assuming that the `'infs'` target-filespec resource for the AppleTalk resource file is 1002, then use the following clause to check for AppleTalk version 53:

Why multiple`'DRVR'` resources for DA installation

Date Written: 9/30/91

Last reviewed: 8/19/92

My Macintosh Installer script installs a desk accessory. Under System 6, each time I run the script, a new copy of the DA appears as a `'DRVR'` resource in the System file. Why?

This happens when the `dontDeleteWhenInstalling` flag is used in conjunction with the `updateExisting` flag. The Installer 3.1 and 3.2 Scripting Guide indicates that resources marked with the `dontDeleteWhenInstalling` flag can be replaced with a new resource. The guide also indicates that the Installer will overwrite a preexisting resource in the target file if the `updateExisting` flag is set. Given these two flag settings, and the use of the replace `byName` (`noByID`) flag, the Installer does not delete the DA. Instead a new `'DRVR'` resource is created with the same name but a new resource ID.

The correct Installer action is accomplished by setting the `deleteWhenInstalling` flag in conjunction with the `updateExisting` flag. On the other hand, use the `dontDeleteWhenInstalling` flag with the `keepExisting` flag.

Including current Mac volume name with reportVolError alert

Date Written: 9/30/91

Last reviewed: 6/14/93

In my Installer script, how can I include the current volume name in a `reportVolError` alert, as many of the installation scripts from Apple do?

The volume name can be included by inserting "^0" as part of the Pascal string passed to the `reportVolError` error-reporting clause.

Macintosh 'incd' resource

Date Written: 9/30/91

Last reviewed: 8/19/92

What is the `'incd'` resource about?

When the MPW `ScriptCheck` tool is used, it reads the script's file creation date/time stamp converting it into a long word with the `Date2Secs` procedure. `ScriptCheck` stores this long word in the `'incd'` resource for use with verifying files when a network installation is performed.

Macintosh Installer preflight checks

Date Written: 9/30/91

Last reviewed: 8/19/92

What checks are made by the Installer when preflighting an installation? Occasionally the alert "Could not find a required file..." occurs and the installation is aborted.

The Installer compiles a list of the source file specifications from each of the resource `'inra'` and file `'infa'` atoms specified among the package 'inpk' atoms included for installation. Each source file specification includes a complete path name. As each source file is accessed, a check is made of the file's creation date/time stamp with the date/time stamp recorded in the corresponding `'infs'` resource. If the date/time stamps do not match, the alert results and the installation is aborted.

Network installation setup considerations

Date Written: 9/30/91

Last reviewed: 8/19/92

What are some of the considerations when configuring a network installation setup?

Under Installer 3.1 or 3.2, network software installations are made possible by setting up an installation folder on the server volume. This folder will contain the Installer application, the Script file, and a folder(s) matching the names of the required disk(s). Within the disk folder(s) are the corresponding contents of the disk(s).

One of the engineering problems that needed to be dealt with can occur when a workstation is used to create the server installation folder. The problem occurs when the system date and time differ significantly between the workstation and the server. Under such a condition, files copied from the workstation to the server may have their creation and modification date time stamps altered. If a modification is made, the "delta" is constant for both the creation and modification date/time stamp and for all files copied at that time.

The Installer preflights a file by comparing its creation date/time stamp with the value stored in the corresponding `'infs'` resource in the script file. To compensate for the fact that a server may alter a file's creation date/time stamp, the Installer implements the `'incd'` resource. After the user selects the Install button, the Installer reads the 'incd' resource and compares it with the script file's creation date/time stamp. The difference is stored as the delta. On a normal disk installation, the delta is always zero. As the Installer finds each required source file, the file's creation date/time stamp is converted to a long word and adjusted by the delta. The modified date/time stamp is then compared with that stored in the script file. If the values match, the file is considered found and the installation proceeds. On network installations, the delta may be nonzero. If so, it indicates that the file's creation date/time stamps were modified when copied to the server. Thus the 'incd' resource gives the Installer a way to maintain file verification even though the date/time stamp may be altered.

A specific problem can occur when an installation is set up on some systems running older versions of Novell server software. Under specific conditions, files copied to some Novell servers have their creation time stamp altered to 12:00 AM regardless of the original time stamp. This includes the creation time stamp of the script file. This condition wreaks havoc with the Installer's preflight mechanism. The delta determined between the `'incd'` resource and the Script file's creation date/time stamp may not be consistent with the creation date/time stamp stored in the `'infs'` resource and the corresponding file's time stamp now at 12:00 AM.

A workaround solution for this problem is to set the creation date/time stamp for all files on the installation disk to 12:00 AM, BEFORE running the `ScriptCheck` tool. Use the MPW tool SetFile to perform this function. Here's a sample MPW script for performing this function:

This script assumes that the current directory is set to the root of the Installation disk. For multiple disks, run this script on each disk.

Installer 3.2 and preserving a file's locked bit

Date Written: 8/28/91

Last reviewed: 8/19/92

When using the Apple Installer 3.2, source files that are locked become unlocked when the Installer copies them to their

destination. The installer preserves some of the other file attribute bits but not the locked bit. How can I preserve the locked bit? Will I have to write an action atom?

The problem here is that the locked bit is a Finder information setting. Were you to have your own Finder comments associated with a file source, you would find that these comments would not be "installed" along with the file.

A solution to the problem is to implement an action atom code resource. Specifically, a sample code resource is now available in the Installer kit available on the latest Developer CD. The code resource takes the target `'infs'` resource as an argument to specify the file on which to set the lock bit. Note that the Installer will abort an installation if an existing file to be replaced is locked. Locking a file means that for a future installation, you will need to tell the user to unlock the file manually before running the installation.

How Macintosh Installer locates source files

Date Written: 8/28/91

Last reviewed: 8/19/92

I notice that if I change the name of the installation disk, the Installer is still able to find the source files. How does this work? I'm writing an action atom code resource and I want to duplicate the source file-finding scheme implemented by the Installer, as described on page 30 of the *Installer 3.1/3.2 Scripting Guide.*

The Installer takes the source file specification and parses off the volume name, leaving the filepath from the root of the source volume. It then searches for the filespec using this parsed file name from the root of the source volume. If the file is not found, the search order continues with the following guidelines:

1. Search for a folder with the disk's name at the same HFS level as the Installer.
2. Search for a folder with the disk's name at the root level of the Installer's volume.
3. Ask for the floppy disk.

In this manner, the Installer is able to install from disk copies of Installer disks when the disk name has been changed. Note that the documentation mentions a fourth guideline, that the Installer searches for a folder with the disk's name at the same HFS level as the script file. This information is incorrect. Neither version 3.2 nor 3.3 implements this guideline.

Macintosh Installer 3.2 makes backup copies of active files

Date Written: 7/10/91

Last reviewed: 8/19/92

If I start up from an 800K floppy disk with only the Macintosh System and Finder files living on it (leaving some 350K of space free), and then try to use Installer 3.2 to move a 130K cdev and an 8K `'adbs'` resource, I get a message that not enough disk space is available on the startup floppy disk. Installer says I need some 505K of space available, yet I am copying less than 150K. Why is this happening?

This results from the "live install" feature of the Installer. Because the floppy disk System file will be written to in this scenario, the Installer first attempts to make a backup copy of the system. This means that there needs to be additional disk space on the startup disk for a copy of all files to be revised--in this case, the System file on the floppy drive. The floppy disk, most likely, does not contain sufficient room to hold a copy of the System file. If the system is started up from a different disk, the Installer installs the new resources into the inactive System file on a floppy disk without creating a backup.

[Back to top](#)

Downloadables

Acrobat version of this Note (K).   [Download](#)

Technical Notes by [Date](#) | [Number](#) | [Technology](#) | [Title](#)
[Developer Documentation](#) | [Technical Q&As](#) | [Development Kits](#) | [Sample Code](#)