Smaller Installer™

An Installer Utility for the Macintosh®

User’s Guide 1.1

Contents

Introduction

Smaller Installer is an installer program created for developers who distribute software, clip art, HyperCard® stacks, databases or other information on floppy disks. It provides a way to distribute data in a compressed format to reduce the number of disks required.

Many developers already use self-extracting archives created by Compact Pro™ for this purpose. Although this is an inexpensive solution, self-extracting archives have several limitations which make them less than ideal for this purpose. Smaller Installer is designed to address some of these limitations. Since Smaller Installer is based on the same compression technology used in Compact Pro, you can get the benefit of Compact Pro's award-winning compression along with an improved user interface which can be customized to meet your specific requirements.

Features

- Significantly reduces the number of disks shipped. Files are stored in compressed form using Compact Pro which typically reduces file size by 50%. The installer application typically requires less than 30K bytes.
- No practical limits on file sizes—individual files may be larger than a single floppy disk. Installer automatically splits files across multiple diskettes as needed. Files are recombined “on the fly” without using any additional disk space on the user’s system.
- Installer dialogs and icons may be customized to display your logo and icons. Customized scrolling text may be added to provide help or instructions.
- Main window displays the disk space required for installation and the space available on the current volume.
- Installation “groups” may be defined to allow the user to install subsets of the distributed files.
- Installed files may be directed to special folders on the user’s disk, e.g. “Control Panels”, “Extensions” or other system-related folders.
- A custom code resource may be executed before and after an installation to perform additional customization functions if necessary.

Licensing/Distribution

Toolkit. The Smaller Installer toolkit is distributed as a self-expanding file named “SI Toolkit Installer”. This package contains a set of tools for building custom installer programs. These tools are provided at no charge. You may freely copy and distribute this toolkit with the following restrictions:

- The toolkit may only be distributed in its original self-expanding format. No partial distributions are allowed. In other words, only the “SI Toolkit Installer” file may be copied and distributed.
- You may not modify the “SI Toolkit Installer” file in any way.

Installers. The custom installers created by Smaller Installer contain a copyrighted software program which must be licensed for distribution. You may create up to 5 copies of an installer for testing and evaluation purposes. Any other use of these copies—including both commercial and noncommercial distribution—is expressly prohibited.

Licenses for distribution of custom installers created by Smaller Installer are available for a one-time fee. This fee authorizes the creation of a specified number of copies based on the following schedule:

Quantity	**License Fee**
1K	$200
10K	$500
Unlimited	$1,000

Note: Prices are subject to change without notice.

Each license covers distribution of a single product. Subsequent releases of the same product do not require additional licenses unless the total number of copies created exceeds the number licensed.

To obtain a distribution license, follow the instructions provided in the file named “SI License Form”.

Contact Information

If you require additional information on technical or licensing issues, please write or call:

US Mail:	Cyclos — SI PO Box 31417 San Francisco, CA 94131-0417
Voice:	415-821-1448
E-mail:	CompuServe — 71101,204 Internet — 71101.204@compuserve.com AppleLink — CYCLOS

Smaller Installer Overview

The Smaller Installer toolkit consists of an installer “builder” application and a set of auxiliary files which are designed to work with the Compact Pro data compression utility. This combination provides a simple way to create a small, customized, multi-disk installer for your product.

Toolkit Contents

The Smaller Installer toolkit contains the following items:

SI Builder—Installer Builder application

SI User’s Guide—User’s manual (in MS Word format)

SI License Form—Installer distribution license form

Language Modules—Folder containing non-English language modules

Starter Resources—Folder containing starter resources for custom dialogs and icons

Hook Proc Examples—Folder containing files to create example hook procedures

Description of Distribution Disk Set

The Builder creates a distribution disk set which consists of one or more disks of any size. The first disk contains an “installer”. If additional disks are required, the Builder stores one “installer data file” on each disk.

You may store any additional files you wish on the first diskette—the Builder will ignore them (a “Read Me First” file is common). All other diskettes are completely erased by the Builder when the disk set is created.

Installer. The installer is an application which interacts with the user and installs your distribution files on the user’s machine. Some or all of your distribution files are imbedded in the installer in a compressed form. The operation of the installer is described more fully in the chapter titled “Installer Operation”.

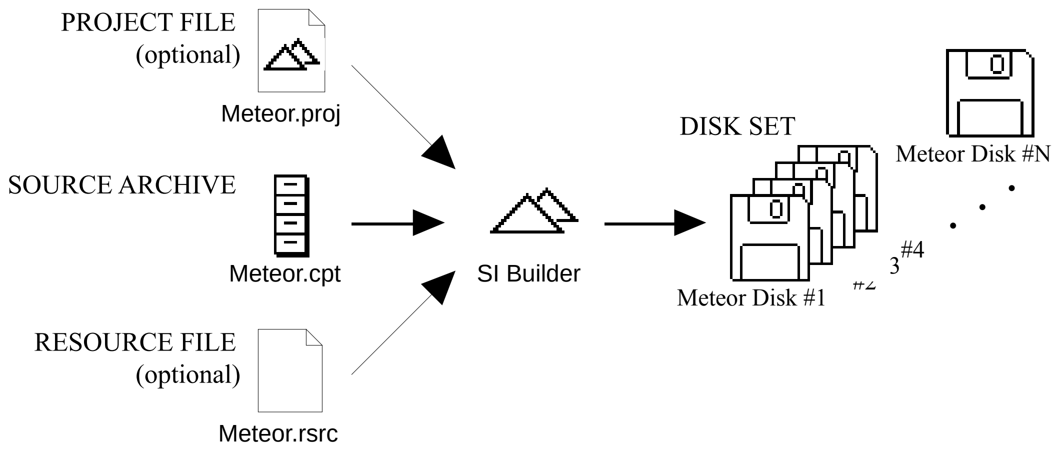
Installer Data Files. The installer data files contain additional blocks of your distribution files in a compressed format which is unique to the Smaller Installer. These files cannot be opened by other applications with meaningful results.

Disk/Installer/Data File Names. You may name the diskettes any way you please as long as the names are unique. Likewise, there are no restrictions on the names you may use for the installer application and the installer data files; however, it is generally a good idea to use unique names for the data files.

Once a disk set has been built, the installer will not work if the disk or data file names are changed. This is due to the installer storing these names internally. To facilitate installations from network servers, the installer will operate properly if the installer application and all data files are stored in the same folder.

Building A Disk Set

To build a disk set, you must first create a “source archive”. If you require customized dialogs or icons, you will also need to create a “resource file”. Next, you must run the Builder application. It uses the source archive and resource file to create all the files in the disk set. As you run the Builder, it saves various configuration information in a project document. You may save this document in a “project file” to simplify subsequent builds of the disk set.



Project File. The project file contains the names and locations of the source archive and resource file as well as various other configuration information used by the Builder. You can create a project file by using the Save or Save As command in the Builder the first time you build a disk set.

Source Archive. The source archive is a Compact Pro archive which contains all the files you wish to distribute. The hierarchical organization and naming of the folders within the source archive specify the rules for installing the files on the user’s system. These rules are described in the chapter titled “Creating The Source Archive”. Note that the source archive is not actually shipped with the disk set—it is only used during the build process to define the data to be shipped.

Resource File. The resource file is a standard Macintosh resource file which you can create using a resource editor such as Resorcerer® or ResEdit™. It contains any customized dialogs or icons which you wish to use in place of the standard dialogs and icons provided by the Builder. The Smaller Installer toolkit includes starter resource files which you may use to create your own customized resources. Additional guidelines for creating installer resources are provided in the chapter titled “Customizing The Installer Resources”.

Installer Operation

The installer application performs the function of decompressing the distribution files and installing them on the user's system. It has a highly customizable user interface which you may tailor to suit your requirements.

Requirements/Limitations

The installer requires System 4.2 or higher and a Macintosh Plus or newer (operation with Macintosh 128K/512K/512KE machines is not supported). It is limited to installing a maximum of 1000 files and folders from one distribution disk set. Disk sets may contain no more than 50 disks. Installation on MFS disks is not supported.

Launching The Installer

The installer must be launched by double-clicking the installer or selecting the installer and using the Open command in the Finder™. The installer purposely inhibits execution when an installer data file is double-clicked. This prevents errors which could be caused by the presence of multiple vendors' installers on the same disk or by multiple versions of one vendor's installer on the same disk.

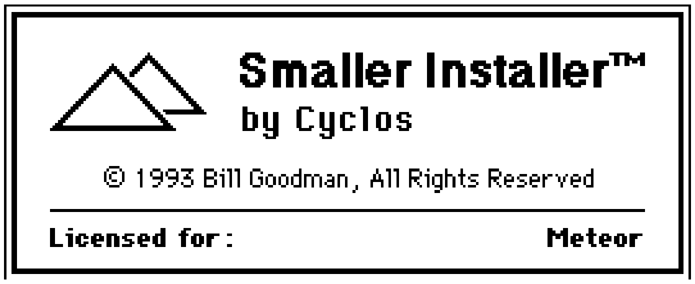

Smaller Installer™
by Cyclos
© 1993 Bill Goodman, All Rights Reserved
Licensed for: Meteor

Figure 1: License Dialog

After launching, the installer displays the License Dialog for several seconds before proceeding to the Main Dialog. You specify the name displayed in the License Dialog by setting the "Product Name" field in the Builder.

Main/AltMain Dialog

The standard version of the Main Dialog is shown in Figure 2. This dialog indicates the target volume. It also shows the disk space required for the installation and the free space available on the target volume. The Drive and Eject buttons may be used to change the target volume. The Install button starts the installation, and the Quit button terminates execution.

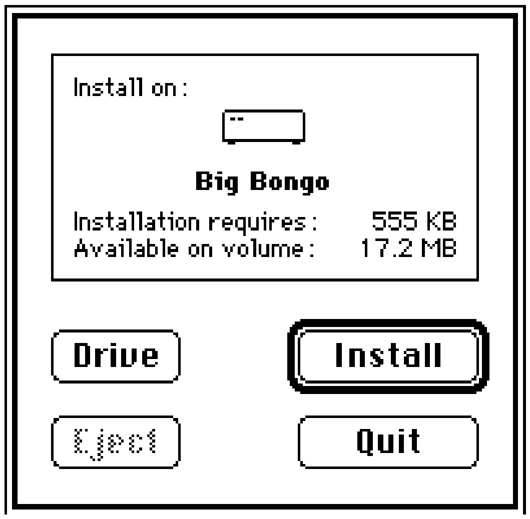
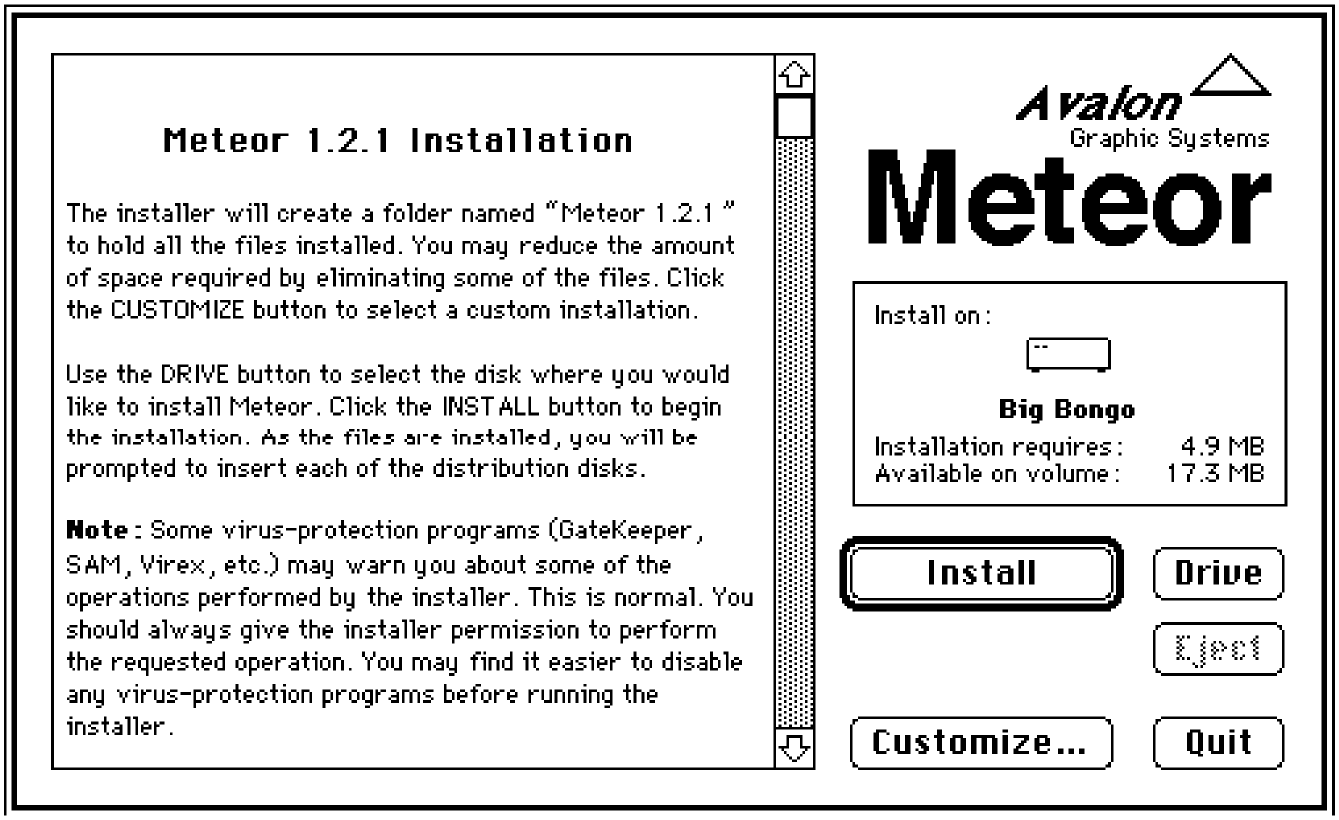


Figure 2: Standard Version of Main Dialog

In addition to the items shown here, the Main Dialog contains a number of additional buttons, checkboxes, radio buttons and user items which are not displayed in the standard dialog. You may modify the Main Dialog to make use of these items or to add custom graphics and text of your own. The items in the dialog may be rearranged and resized in a variety of ways.

Figure 3 shows a version of the Main Dialog which has been customized by adding a company’s logo and help text. The logo requires the addition of one ‘PICT’ resource and the scrolling help text requires one ‘TEXT’ and one ‘styl’ resource.


Meteor 1.2.1 Installation

The installer will create a folder named "Meteor 1.2.1" to hold all the files installed. You may reduce the amount of space required by eliminating some of the files. Click the CUSTOMIZE button to select a custom installation.

Use the DRIVE button to select the disk where you would like to install Meteor. Click the INSTALL button to begin the installation. As the files are installed, you will be prompted to insert each of the distribution disks.

Note : Some virus-protection programs (GateKeeper, SAM, Virex, etc.) may warn you about some of the operations performed by the installer. This is normal. You should always give the installer permission to perform the requested operation. You may find it easier to disable any virus-protection programs before running the installer.

Install on :

Big Bongo

Installation requires : 4.9 MB
Available on volume : 17.3 MB

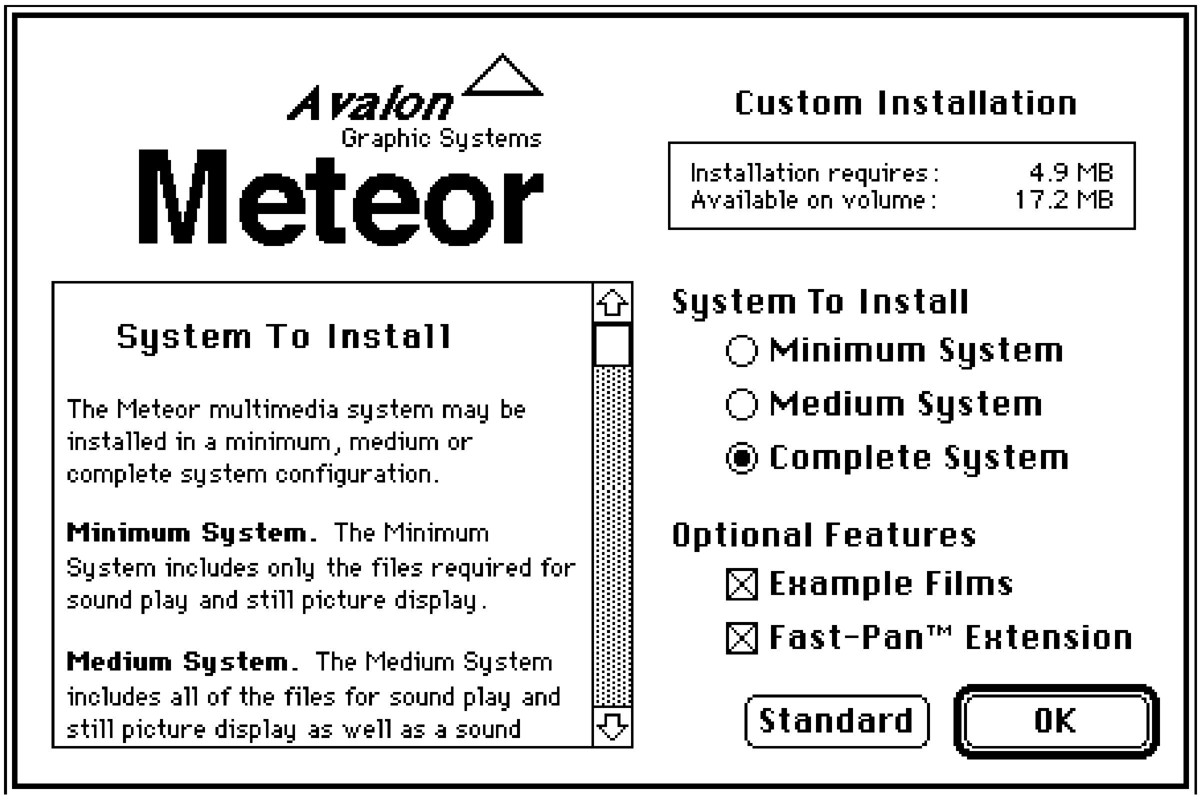
Figure 3: Customized Version of Main Dialog

AltMain Dialog. In addition to the Main Dialog, the installer can display an alternate dialog which is functionally similar to the Main Dialog. This dialog is called the AltMain Dialog. The AltMain Dialog may be used for any purpose; however, its most common use is as a "Customize" dialog to allow the user to modify the installation parameters. It can also be used as a separate Help dialog.

All of the items in the AltMain Dialog operate exactly the same as the items in the Main Dialog with two exceptions:

- The Install button is changed to an OK button. It performs the same function as the Switch button described below.
- The scrolling help text uses different resources so the AltMain Dialog can display scrolling text which is different from the text displayed in the Main Dialog.

Switch Button. The Main and AltMain Dialogs both contain a Switch button. The Switch button dismisses the currently displayed dialog and switches to the alternate dialog (i.e. the Main Dialog switches to the AltMain Dialog and the AltMain Dialog switches to the Main Dialog). In Figure 3, the Switch button has been renamed “Customize...”. When the Customize button is clicked in this example, an AltMain Dialog like the one in Figure 4 could be displayed.



System To Install

The Meteor multimedia system may be installed in a minimum, medium or complete system configuration.

Minimum System. The Minimum System includes only the files required for sound play and still picture display.

Medium System. The Medium System includes all of the files for sound play and still picture display as well as a sound

Installation requires: 4.9 MB
Available on volume: 17.2 MB

System To Install
○ Minimum System
○ Medium System
◉ Complete System

Optional Features
☒ Example Films
☒ Fast-Pan™ Extension

Figure 4: AltMain Dialog Configured as a “Customize” Dialog

Group Checkboxes and Radio Buttons. The Main and AltMain Dialogs contain 16 pre-defined checkboxes and 10 radio buttons which may be used to define “installation groups”. Installation groups are subsets of the files in the distribution disk set. By including some of these checkboxes and radio buttons in your dialogs and defining special folders in the source archive, you can provide a way for users to tailor the installation to their own needs. The AltMain Dialog in Figure 4 uses three of the group radio buttons and two of the group checkboxes to provide installation options. Additional information on using installation groups is provided in the “Group Folders” section of the chapter titled “Creating The Source Archive”.

Standard Button. The Standard button sets all the group checkboxes and group radio buttons to a default setting which you define using the “Standard Groups” command in the Builder.

Installing The Files

When the user clicks the Install button, the installer begins expanding the distribution files and stores them on the target volume selected by the user. The status area of the Main Dialog changes to display installation progress as in Figure 5.

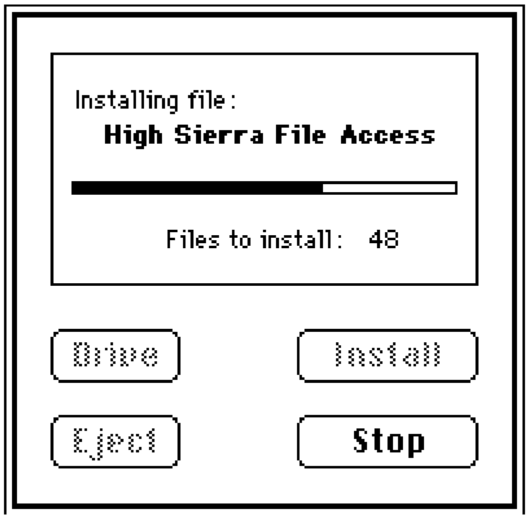

Installing file:
High Sierra File Access
Files to install: 48

Figure 5: Main Dialog During File Installation

During the installation process, the installer prompts the user to insert the various distribution disks as needed using the dialog in Figure 6.

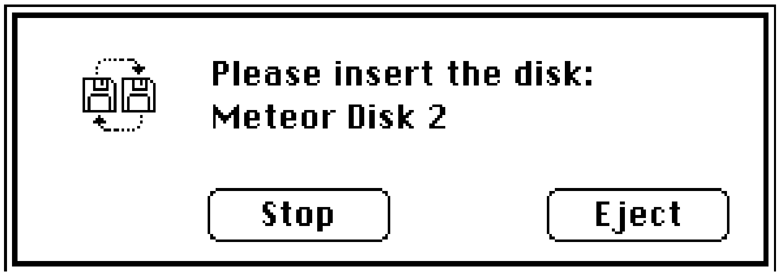
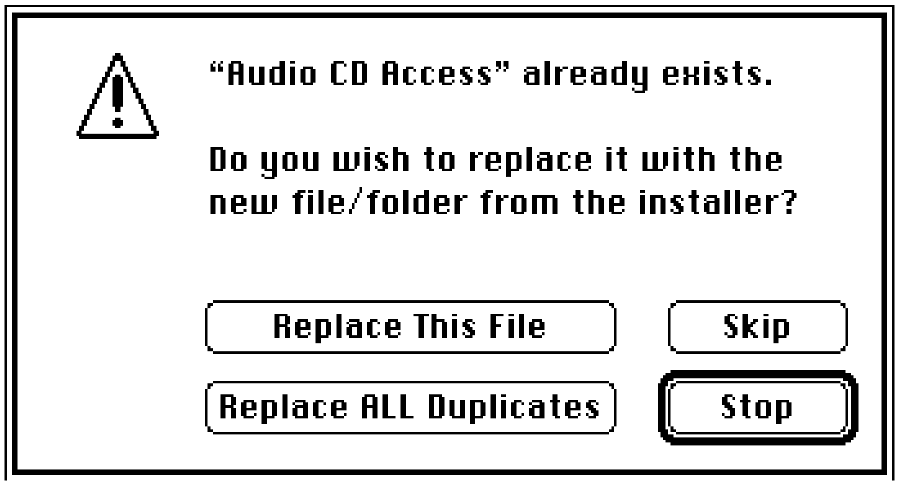

Please insert the disk:
Meteor Disk 2

Figure 6: Disk Request Dialog

If the installer finds that one of the files in the distribution set already exists on the target volume, it displays the dialog in Figure 7 to allow the user to decide the proper action.


"Audio CD Access" already exists.

Do you wish to replace it with the new file/folder from the installer?

Figure 7: Replace Dialog

You can force the installer to replace any pre-existing files without asking the user by checking the "Replace files without asking" checkbox in the Builder.

Completion Dialog

After all selected files have been installed, the installer displays a Completion Dialog. The standard dialog is shown in Figure 8.

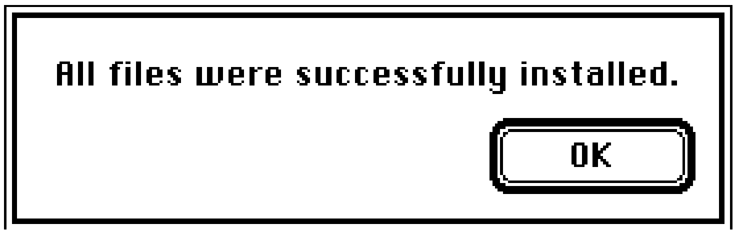
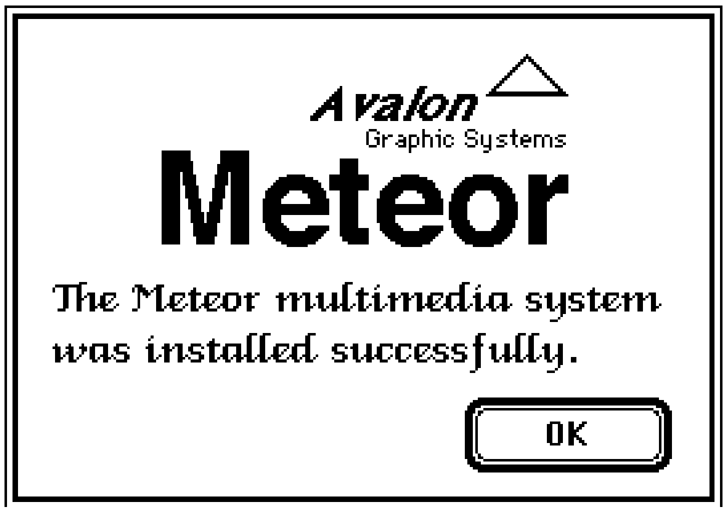

All files were successfully installed.

Figure 8: Standard Version of Completion Dialog

This dialog may be customized by adding text or pictures as demonstrated in Figure 9.



The Meteor multimedia system was installed successfully.

Figure 9: Customized Version of the Completion Dialog

The installer terminates execution when the Completion Dialog is dismissed. Note that the installer does not display this dialog or terminate execution if any of the files were skipped during installation.

Installer Hook Procedure

You may customize the operation of the installer by including a code resource called an “Installer Hook Procedure”. If you supply a hook procedure, the installer will call it at various points during the installation to allow you to customize the installation process.

A hook procedure may be used to test for the presence of required hardware or software. It can modify the set of files to be installed by enabling or disabling various installation groups. It can also perform various functions after the files have been installed—such as personalizing a newly installed application or deleting unnecessary files from a previous version of the application.

Additional information on installer hook procedures is provided in the chapter titled “Using An Installer Hook Procedure”.

Creating The Source Archive

The “source archive” is a Compact Pro archive which contains all the files you wish to distribute with your custom installer. The hierarchical organization and naming of the folders in the archive determine the way the files will be installed on the user’s system. This chapter describes the rules for organizing the archive. If you require additional information on using the Compact Pro utility to create the archive, please refer to the Compact Pro User’s Guide provided with the Compact Pro package.

Standard Folders

The installer reproduces the folder hierarchy of the source archive on the user’s system beginning in the root folder of the target volume selected by the user. Any folders defined in the source archive which do not already exist on the target volume will be created by the installer.

System-related Folders

The following special folder names have been defined to refer to various system-related folders:

$SYSTEM	Currently active system folder
$APPLE_MENU	Apple Menu Items folder
$CONTROL_PANELS	Control Panels folder
$EXTENSIONS	Extensions folder
$PREFERENCES	Preferences folder
$STARTUP	Startup Items folder

If a folder in the source archive has one of these special names, the contents of that folder will be placed in the specified system-related folder. Any folder hierarchy within the source archive folder will be reproduced inside the system-related folder on the user’s system.

Under System 7, any system-related folder will automatically be created if it does not already exist. Under System 6, all of these folders are directed to the currently active system folder—no folders are ever created.

Note that the installer always installs these files in the active system folder even if that folder is not on the same volume as the target volume selected by the user.

Group Folders

Installation groups allow users to install subsets of the files in the distribution disk set. You define the contents of the groups by creating “group folders” in the source archive. At installation time, the user may select which groups will be installed by clicking on checkboxes and radio buttons which you have defined in the installer dialogs.

The installer supports 26 groups which are designated by the letters A-Z. The first 16 groups (A-P) are associated with checkboxes in the installer dialogs, so they may be selected in any combination. The next five groups (Q-U) are associated with a set of radio buttons, so only one of these may be selected at a time. Likewise, the final five groups (V-Z) are associated with a second set of radio buttons.

A group folder is defined by starting a folder name with a ‘{‘ character. This is followed by one or more letters designating a group. Comment text may be added to a group folder name after a closing ‘}’ character.

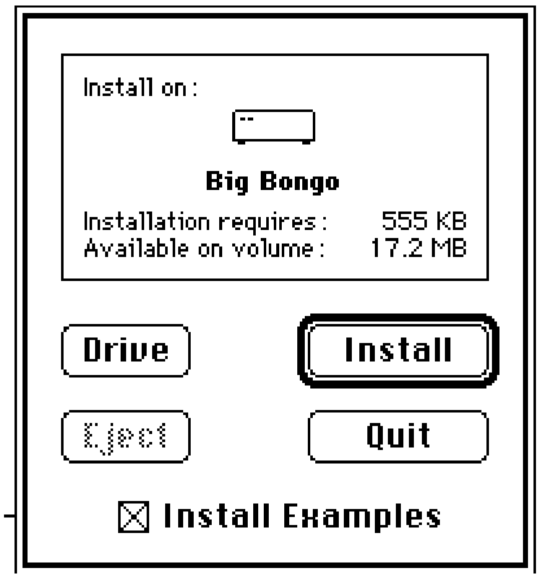
For example, a folder named “{A} Color Files” defines a group folder for group A (“Color Files” is a descriptive comment). The contents of this folder will only be installed if the user selects the group A checkbox. Note that the group folder itself will NOT be created on the target volume—only the contents of the folder will be created.

If more than one group is specified in a group folder name, the folder will be installed if ANY of the groups in the list are selected. For example, the contents of a folder named “{ABQ}” would be installed if either checkbox A, checkbox B or radio button Q were selected by the user. This provides a logical OR’ing function for installation groups.

A logical AND’ing function may be obtained by placing one group folder inside another. For example, if a folder named “{J}” is created inside another folder named “{K}”, the installer will only install the contents of the first folder if BOTH checkbox J and checkbox K are selected by the user.

Examples

Example #1. This example shows the list of files in a source archive for an installer which has one installation group. The group A checkbox in the Main Dialog is labeled "Install Examples".

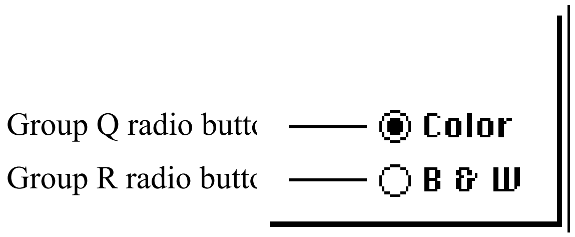


Group A checkbox

Meteor Folder
    Meteor
    Meteor Help
    {A} example files
        Examples Folder
            Example 1
            Example 2
$PREFERENCES
    Meteor Preferences

This installer will create a folder named "Meteor Folder" on the target volume and store the "Meteor" and "Meteor Help" files in the folder. It will also store the "Meteor Preferences" file in the "Preferences" folder inside the user's system folder. If the user has checked the "Install Examples" checkbox, the installer will also create a folder named "Examples Folder" inside the "Meteor Folder". "Example 1" and "Example 2" will be stored in the "Examples Folder".

Example #2. This example shows how one of two groups can be installed in the same folder. The installer will create a folder named "Meteor Folder" on the target volume and store the “Meteor” file in the folder. If the user selects the “Color” radio button, the installer will add the “Color Plug-In” file to the Meteor Folder and store the “Color Extension” file in the Extensions folder. If the user selects the “B & W” radio button, the installer will add the “B&W Plug-In” file to the Meteor Folder and store the “B&W Extension” file in the Extensions folder.



Meteor Folder
    Meteor
{Q} color files
    Meteor Folder
        Color Plug-In
    $EXTENSIONS
        Color Extension
{R} b&w files
    Meteor Folder
        B&W Plug-In
    $EXTENSIONS
        B&W Extension

Note that this organization keeps all of the files associated with a color installation in one place in the source archive and all of the files associated with a black-and-white installation in another. An alternative organization would be:

Meteor Folder
    Meteor
    {Q} color files
        Color Plug-In
    {R} b&w files
        B&W Plug-In
$EXTENSIONS
    {Q} color files
        Color Extension
    {R} b&w files
        B&W Extension

Either arrangement is acceptable. The former tends to be easier to maintain if you have a large number of files in an installation group. The latter may be simpler if the installation groups only affect a few files.

Customizing The Installer Resources

All the customized resources for your custom installer are stored in a single resource file. When the Builder creates an installer, it checks this file to see if you have provided customized versions of any of the standard installer resources. If it finds a customized resource, it replaces the standard installer resource with your customized version. It also copies any other resources you've created to the installer's resource fork.

Resource IDs. The installer uses resource IDs 1000 and higher to indicate resources which must not be modified. Since these resources are copyrighted, the Smaller Installer distribution license specifically prohibits the modification of any of these resources. To prevent conflicts, you should use IDs below 1000 for all your resources.

Resource Attributes. Except as noted, all resources should be created with none of the resource attributes asserted, i.e. resources should be non-Preloaded, non-Purgeable, un-Locked, un-Protected and non-SysHeap.

General Limitations On Customizing Dialogs. When customizing dialogs, you may generally move, resize and rename any item except where noted in the following sections. You may also add 'PICT' and static text items as desired. Do NOT delete unused dialog items. If you do not use an item, you should move it outside the visible area of the dialog window. Dialog items must not be renumbered, and the dialog window types must not be changed.

Limitation On PICT Size. The installer is preconfigured with the assumption that no PICT resource is larger than 50K. If any of your PICT resources are larger than this, you must manually increase the memory allocation in the SIZE resource of the final installer to reflect the increased requirement.

Starter Resources

The Smaller Installer toolkit includes the following starter resource files in the “Starter Resources” folder:

Standard Dialogs.rsrc—Standard dialog resources created by the Builder

Standard Icons.rsrc—Standard icon resources created by the Builder

Basic Example.rsrc—Example using a Main Dialog with a PICT containing a company logo

Text/Group Example.rsrc—Example using a Main Dialog with a help text field and one group checkbox

AltMain Example.rsrc—Example using an AltMain Dialog as a “Customize” dialog and a customized Completion Dialog

These files may be used as a starting point for creating your own customized dialogs and icons.

Customizing The Main Dialog

The Main Dialog requires the following resources:

Type	ID	Attributes	Description
‘DLOG’	300	Purgeable	Main Dialog template
‘DITL’	300	Purgeable	Main Dialog item list

Any ‘PICT’ or ‘ICON’ resources added to this dialog must be purgeable. The visible attribute in the ‘DLOG’ resource should be FALSE.

Item #1: Install Button. This button initiates the installation process. A bold outline is drawn around this button while the installer is idle.

Item #2: Quit Button. This button terminates execution of the installer.

Item #3: Switch Button. This button dismisses the Main Dialog and displays the AltMain Dialog. If you are not using the AltMain Dialog, you should move this button outside the visible area of the Main Dialog.

Item #4: Standard Button. This button sets the group checkboxes and radio buttons to their default state. If you are not using the group checkboxes and radio buttons in the Main Dialog, you should move this button outside the visible area.

Item #5: Drive Button. This button changes the volume selected for installation.

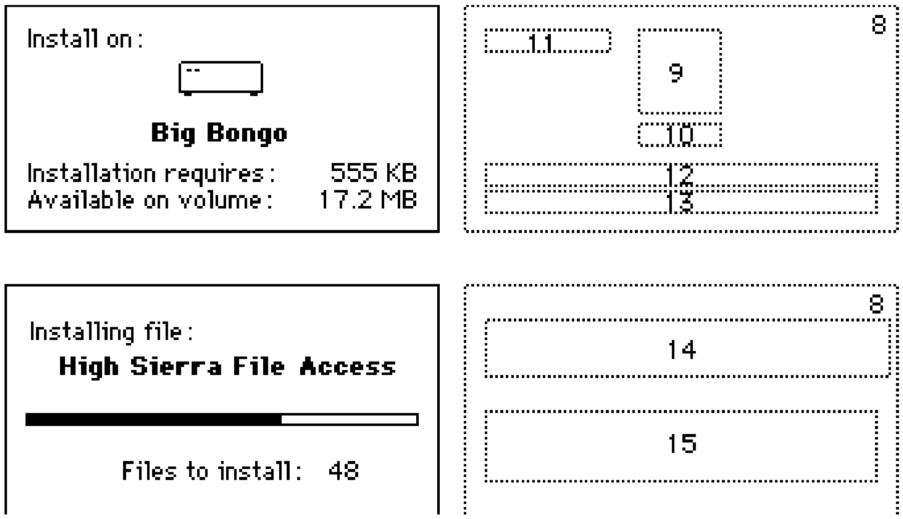
Item #6: Eject Button. This button unmounts and ejects the currently selected volume.

Item #7: Help User Item. This item specifies the location where the scrolling help text (if any) will be displayed. To define this text, you must create the following resources:

Type	ID	Attributes	Description
'TEXT'	300		Main help text (32K bytes maximum)
'styl'	300		Main help style definition (optional)

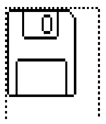
Do not set the height or width of this rectangle to 0. If you do not use this item, move it outside the dialog's visible area and delete the 'TEXT' and 'styl' resources.

Item #8: Status User Item. This item specifies the location of the installer's status display area. All visible user items except the Help User Item must be placed inside this rectangle.



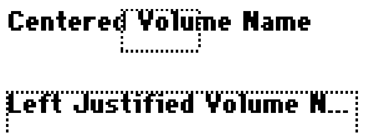
Status Display Area User Items

Item #9: Volume Icon User Item. This item specifies the location of the target volume icon. The item is only drawn when the installer is idle.



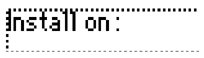
Item #9: Volume Icon User Item

Item #10: Volume Name User Item. This item specifies the location of the target volume name. If this item is less than 36 pixels wide, the volume name will be centered at the top of the rectangle. If it is 36 or more pixels wide, the volume name will be left justified at the top of the rectangle and clipped to the rectangle boundary. This item is only displayed while the installer is idle.



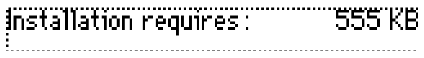
Item #10: Volume Name User Item

Item #11: Install Prompt User Item. This item specifies the location of the "Install on:" string. This item is only displayed while the installer is idle.



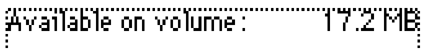
Item #11: Install Prompt User Item

Item #12: Space Required User Item. This item specifies the location of the "Installation requires:" string. This item is only displayed while the installer is idle.



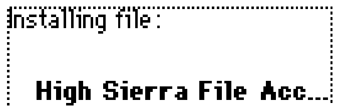
Item #12: Space Required User Item

Item #13: Space Available User Item. This item specifies the location of the "Available on volume:" string. This item is only displayed while the installer is idle.



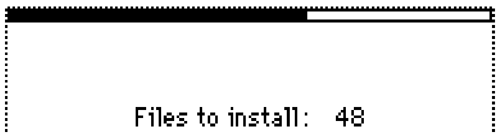
Item #13: Space Available User Item

Item #14: Filename User Item. This item specifies the location of the "Installing file:" string and the filename string. The filename is inset 12 pixels from the left and displayed at the bottom of the rectangle. This item is only displayed while the installer is installing files.



Item #14: Filename User Item

Item #15: Progress Bar User Item. This item specifies the location of the progress bar and the "Files to install: nnn" string. The files-to-install string is centered at the bottom of the rectangle. This item is only displayed while the installer is installing files.



Item #15: Progress Bar User Item

Item #16-31: Group A-P Checkboxes. These checkboxes enable installation groups A through P. The checkbox titles may be changed as desired, but the item numbers must not be modified.

Item #:	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Group:	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P

Item #32-36: Group Q-U Radio Buttons. These radio buttons operate as a set and enable installation groups Q through U. The radio button titles may be changed as desired, but the item numbers must not be modified.

Item #:	32	33	34	35	36
Group:	Q	R	S	T	U

Item #37-41: Group V-Z Radio Buttons. These radio buttons operate as a set and enable installation groups V through Z. The radio button titles may be changed as desired, but the item numbers must not be modified.

Item #:	37	38	39	40	41
Group:	V	W	X	Y	Z

Customizing The AltMain Dialog

The installer displays the AltMain Dialog when the “Switch” button in the Main Dialog is clicked. This dialog requires the following resources:

Type	ID	Attributes	Description
‘DLOG’	301	Purgeable	AltMain Dialog template
‘DITL’	301	Purgeable	AltMain Dialog item list

Any ‘PICT’ or ‘ICON’ resources added to this dialog must be purgeable. The visible attribute in the ‘DLOG’ resource should be FALSE.

All items in the AltMain Dialog operate identically to the items in the Main Dialog with the following exceptions.

Item #1: OK Button. This button dismisses the AltMain Dialog and displays the Main Dialog. A bold outline is drawn around this button while the installer is idle. This button replaces the “Install” button in the Main Dialog.

Item #7: Help User Item. This item specifies the location where the scrolling help text (if any) will be displayed. To define this text, you must create the following resources:

Type	ID	Attributes	Description
‘TEXT’	301		AltMain help text (32K bytes maximum)
‘styl’	301		AltMain help style definition (optional)

Do not set the height or width of this rectangle to 0. If you do not use this item, move it outside the dialog’s visible area and delete the ‘TEXT’ and ‘styl’ resources.

Customizing The Completion Dialog

The installer displays the Completion Dialog after all files have been installed. This dialog requires the following resources:

Type	ID	Attributes	Description
‘DLOG’	302	Purgeable	Completion Dialog template
‘DITL’	302	Purgeable	Completion Dialog item list

Any ‘PICT’ or ‘ICON’ resources added to this dialog must be purgeable.

Item #1: OK Button. This button dismisses the Completion Dialog and terminates execution of the installer.

The installer will dismiss the Completion Dialog if any enabled item is hit. One common configuration is to move the OK button outside the dialog window and fill the screen with a large ‘PICT’ item. If the ‘PICT’ item is enabled, the dialog will be dismissed when the user clicks in the ‘PICT’ or when the RETURN key is pressed.

Customizing The File Icons

The standard installer includes two sets of icons—one set for the installer application and a second set for the installer data files. To replace the standard icons with your own custom icons, follow these steps.

First, select a unique creator code to be used as the signature for your installer. This code MUST be different from the signature used by the standard installer (‘SIAP’), and it MUST be registered with Apple DTS to ensure that it does not conflict with any other applications. Enter this code in the “Creator Code” field of the Builder project document before you build your disk set.

Second, create a set of icon resources and add them to your installer resource file. As a minimum, you must create the following icon resources:

Type	ID	Attributes	Description
‘ICN#’	200		Application file icon and mask
‘ICN#’	201		Installer data file icon and mask

To support color icons, you must also create these resources:

Type	ID	Attributes	Description
‘icl4’	200		Large 4-bit color application file icon
‘icl4’	201		Large 4-bit color installer data file icon
‘icl8’	200		Large 8-bit color application file icon
‘icl8’	201		Large 8-bit color installer data file icon
‘ics#’	200		Small black/white application file icon
‘ics#’	201		Small black/white installer data file icon
‘ics4’	200		Small 4-bit color application file icon
‘ics4’	201		Small 4-bit color installer data file icon
‘ics8’	200		Small 8-bit color application file icon
‘ics8’	201		Small 8-bit color installer data file icon

Using An Installer Hook Procedure

You may augment the operation of your custom installer by supplying an “Installer Hook Procedure”. A hook procedure is a code resource which is stored in the installer’s resource fork. The installer calls the hook procedure at startup, whenever the target volume is changed, before the actual installation begins and after all files have been installed. This provides a number of opportunities to customize the installation process. Some typical uses for a hook procedure are:

- Check for specific machine features before allowing installation to proceed, e.g. minimum operating system version
- Automatically select the files to install based on specific machine features, e.g. install color files only on systems with color monitors
- Personalize an application after it has been successfully installed
- Update or remove files from a previous installation of an older version of an application

To create a hook procedure, use an application development system such as MPW or THINK C™ to create a “code resource” with the following definition:

Type	ID	Attributes	Description
‘SICR’	500		Smaller Installer code resource

Add this resource and any supplemental resources it requires (such as ‘ALRT’s or ‘DITL’s) to the installer resource file. All resource IDs must be less than 1000, and they must not conflict with any of the predefined resources used by the installer.

Functional Description

When the installer is launched, it loads the hook procedure code resource into high memory and locks it in place. The hook remains there until the installer terminates.

The hook procedure code resource has a single entry point with the following Pascal definition:

```
        PROCEDURE SIHookProc            (VAR parmBlkPtr: SIHookParmBlk);
```

The installer calls this procedure to perform three different functions: SetTargetVol, BeginInstall and EndInstall. These functions and the parameter block are described in the following sections.

Global Variables. Because the installer hook procedure is a code resource, it must use A4-based addressing to access any variables declared “global” in the hook procedure. Since the code resource remains locked the entire time the installer is running, it is safe to assume that the “global” variables will be nonvolatile.

Heap Usage. The installer is preconfigured to reserve 10K of heap space for the hook procedure code resource and any other non-purgeable resources the hook procedure may require. In addition, the hook procedure can always assume that 50K of heap space is available for purgeable resources. This space may also be used for non-purgeable resources as long as they are released before the hook procedure returns to the installer.

If your hook procedure requires additional heap space, you can manually increase the ‘SIZE’ resource in your custom installer after each build, or you may add a ‘SIZE’ resource to your resource file to automatically set the memory partition size.

Parameter Block Definition

The hook procedure parameter block has the following structure:

```
CONST  { Function selector }
       siHookSetTargetVol      = 1;          { Set target volume }
       siHookBeginInstall      = 2;          { Begin install }
       siHookEndInstall        = 3;          { End install }

       { Completion Status }
       siHookComplete          = 0;          { Install completed without error }
       siHookFileSkipped       = 1;          { Install completed but some files were
                                               skipped }
       siHookAborted           = 2;          { Install was aborted due to errors or
                                               user cancellation }
       { Result codes }
       siHookNoErr             = 0;          { No error }
       siHookAbort             = 1;          { Error - abort installation }

TYPE   SIHookParmBlk           =
       RECORD
       function:               INTEGER;      { Function selector }
       targetVRefNum:          INTEGER;      { VRefNum of target volume }
       groupAPFlags:           INTEGER;      { Group A-P flags (MSB:P -- LSB:A) }
       groupQUSel:             INTEGER;      { Group Q-U selector (0:Q -- 4:U) }
       groupVZSel:             INTEGER;      { Group V-Z selector (0:V -- 4:Z) }
       completionSts:          INTEGER;      { Completion status for EndInstall }
       result:                 INTEGER;      { Returned result }
       END;
```

“function” indicates the reason the hook procedure was called.

siHookSetTargetVol	Target volume was initialized due to installer launch or user has changed target volume
siHookBeginInstall	User has clicked INSTALL button
siHookEndInstall	Installation has completed

“targetVRefNum” specifies the volume reference number of the target volume selected by the user.

“groupAPFlags” indicates which of the installation groups ‘A’ through ‘P’ are selected for installation. Each bit corresponds to a group—the least significant bit corresponds to group ‘A’ and the most significant bit corresponds to group ‘P’. A ‘1’ in a bit position indicates the group will be installed and a ‘0’ indicates it will not be installed.

“groupQUSel” indicates which one of the installation groups ‘Q’ through ‘U’ is selected for installation. A value of 0 corresponds to group ‘Q’ and a value of 4 corresponds to group ‘U’.

“groupVZSel” indicates which one of the installation groups ‘V’ through ‘Z’ is selected for installation. A value of 0 corresponds to group ‘V’ and a value of 4 corresponds to group ‘Z’.

“completionSts” indicates the completion status at the end of an installation. It is only valid for the EndInstall function.

siHookComplete	Installation completed without error and all selected files were installed
siHookFileSkipped	Installation completed but some selected files were skipped
siHookAborted	Installation was aborted due to errors or because the user cancelled

“result” is the result code returned by the hook procedure for BeginInstall and EndInstall function calls.

siHookNoErr	No error—proceed with installation
siHookAbort	Abort installation

SetTargetVol Function

Input Parameters:

function	siHookSetTargetVol
targetVRefNum	Target volume reference number
groupAPFlags	Group A-P flags
groupQUSel	Group Q-U selector
groupVZSel	Group V-Z selector

Returned Values:

groupAPFlags	Group A-P flags
groupQUSel	Group Q-U selector
groupVZSel	Group V-Z selector

The installer executes the SetTargetVol function whenever the target volume is changed. This occurs when the initial target volume is selected (at installer launch) and each time the user changes the target volume. The group parameters indicate the groups which are currently selected to be installed. SetTargetVol can modify the groups selected for installation by changing the group parameters before returning.

BeginInstall Function

Input Parameters:

function	siHookBeginInstall
targetVRefNum	Target volume reference number
groupAPFlags	Group A-P flags
groupQUSel	Group Q-U selector
groupVZSel	Group V-Z selector

Returned Values:

result	Hook result

The installer executes the BeginInstall function when the user clicks the INSTALL button to begin an installation. BeginInstall may force the installation to be aborted by returning “siHookAbort” in “result”.

EndInstall Function

Input Parameters:

function	siHookBeginInstall
targetVRefNum	Target volume reference number
groupAPFlags	Group A-P flags
groupQUSel	Group Q-U selector
groupVZSel	Group V-Z selector
completionSts	Completion status

Returned Values:

result	Hook result

The installer executes the EndInstall function whenever an installation completes. “completionSts” is set to indicate the completion status of the installation. EndInstall may force the installation to be aborted by returning “siHookAbort” in “result”.

If all files were successfully installed, the installer calls EndInstall with “completionSts” set to “siHookComplete”. This call occurs before the installer displays the Completion Dialog, so the hook procedure can still abort the installation if a problem is detected.

If some files were skipped during the installation—either due to file errors or due to the user refusing to replace pre-existing files—the installer calls EndInstall with “completionSts” set to “siHookFileSkipped”.

If an unrecoverable error occurs or the user stops the installation after the BeginInstall function has been executed, the installer calls EndInstall with “completionSts” set to “siHookAborted”.

Hook Procedure Examples

The Smaller Installer toolkit includes the source code for several example hook procedures in the “Hook Proc Examples” folder. This folder also contains a common interface header file named “SIHookProc.h”.

DemoHook. DemoHook is a hook procedure which displays an alert each time the hook procedure is called. The “DemoHook” folder contains the following files:

DemoHook.c—C source code for hook procedure

DemoHook.π—THINK C™ project file

DemoHook.Misc.rsrc—Resources used by the hook procedure

DemoHook.SICR.rsrc—Compiled hook procedure code resource

Sys6Or7Hook. Sys6Or7Hook is a hook procedure which checks the version of the System that is running and enables group "Q" if System 7 is running or group "R" if System 6 is running. The “Sys6Or7Hook” folder contains the following files:

Sys6Or7Hook.c—C source code for hook procedure

Sys6Or7Hook.π—THINK C™ project file

Sys6Or7Hook.SICR.rsrc—Compiled hook procedure code resource

QD32Hook. QD32Hook is a hook procedure which checks that the target machine is running System 6.0.5 or higher. It also enables or disables an installation group to install the 32-bit Quickdraw file if the target machine is not running version 1.2 or higher of Quickdraw. The “QD32Hook” folder contains the following files:

QD32Hook.c—C source code for hook procedure

QD32Hook.π—THINK C™ project file

QD32Hook.Misc.rsrc—Resources used by the hook procedure

QD32Hook.SICR.rsrc—Compiled hook procedure code resource

PersonalizeHook. PersonalizeHook is a hook procedure which personalizes an installed application. It prompts the user to enter name and organization strings then saves these strings in an invisible file on the distribution disk. After all files have been installed, PersonalizeHook adds these strings to a resource in the installed application. The “PersonalizeHook” folder contains the following files:

PersonalizeHook.c—C source code for hook procedure

PersonalizeHook.π—THINK C™ project file

PersonalizeHook.Misc.rsrc—Resources used by the hook procedure

PersonalizeHook.SICR.rsrc—Compiled hook procedure code resource

Volume Info—Example invisible file

Working With The Builder

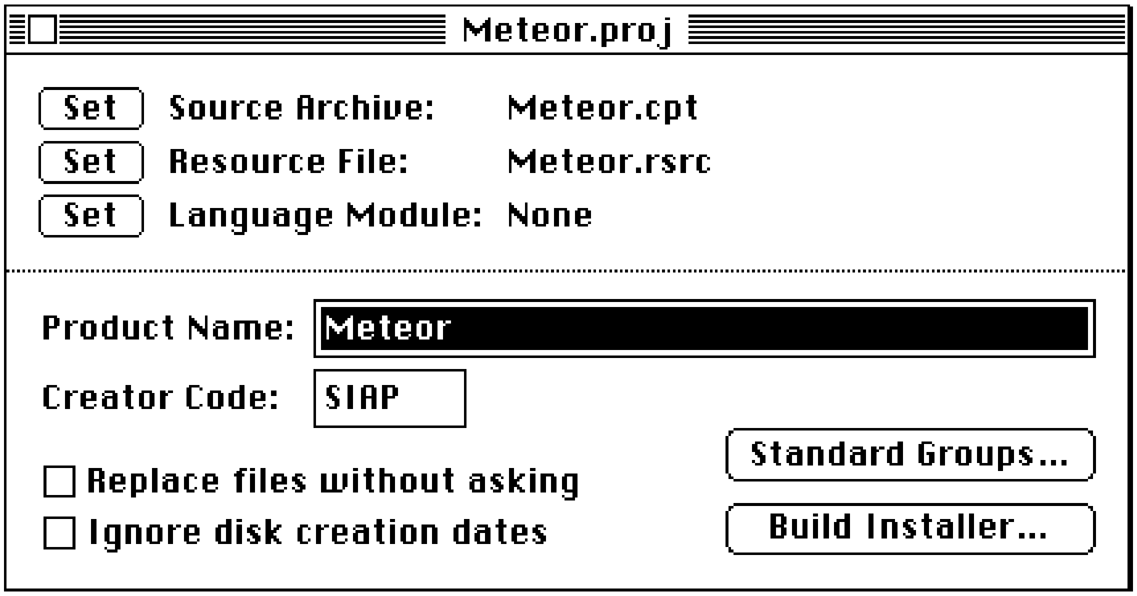
The SI Builder is an application which creates distribution disk sets. It combines the data from a source archive and an optional resource file to create a custom installer and a set of installer data files.

Requirements/Limitations

The Builder requires System 7.0 or higher and a Macintosh Plus or newer.

The Project File

Before you can build an installer, you must specify various configuration information to be used by the Builder. The Builder saves this information in a "project" document like the one in Figure 10.



Meteor.proj

Source Archive: Meteor.cpt
Resource File: Meteor.rsrc
Language Module: None

Product Name: Meteor
Creator Code: SIAP

☐ Replace files without asking
☐ Ignore disk creation dates

Figure 10: Project Document

To create a new project document, use the New command in the File menu. You may save the project document for future builds by using the Save or Save As commands in the File menu. By convention, project files are named with a ".proj" extension; however, any name is acceptable.

Set Source Archive

Use the “Set Source Archive” button to specify the Compact Pro source archive used to build your disk set. The source archive must contain only one segment and it must not be self-extracting. For best results, the source archive should be stored on a hard disk.

Set Resource File

Use the “Set Resource File” button to specify the name of a resource file containing your customized installer resources. This file is optional.

Set Language Module

The Builder can create installers in several different languages. To select a language other than English, use the “Set Language Module” button to select one of the modules in the “Language Modules” folder. If no module is selected, the Builder creates an English-language installer.

Product Name

Enter the name of your product in the “Product Name” field. This name will be displayed in the “Licensed for:” field of the License Dialog each time the installer is launched. The Builder allows up to 255 characters for the product name; however, only a single line is displayed in the License Dialog.

Note: This name is defined at the time you pay your license fee. Since it cannot be changed for future releases, you should not include a version number as part of the name.

Creator Code

The “Creator Code” field defines the 4 character creator code which will be used to build the disk set. This creator code is assigned to the installer application and installer data files created by the Builder. It is also used to create the bundle resources in the installer application.

The default value for this code is ‘SIAP’. If you do not require custom icons for your installer, you should use the standard creator code. If you plan to add custom icons, you must select a different creator code and enter it here. Note that your new creator code must be registered with Apple DTS to ensure that it does not conflict with any existing or future applications.

Replace Files Without Asking

During installation, the installer normally prompts the user with the dialog in Figure 7 whenever a pre-existing file is encountered. Check the “Replace files without asking” checkbox to have the installer overwrite all pre-existing files without prompting the user.

Ignore Disk Creation Dates

The installer identifies the disks in a disk set by remembering each disk’s name and creation date. This can cause problems for users who wish to copy your disks by dragging the installer files from one disk to another since the creation date will not be correct. If you wish to make it easier for users to duplicate your disks, check the “Ignore disk creation dates”. This prevents the installer from checking the creation dates on disks.

Standard Groups

Use the “Standard Groups” button to set the default group configuration for the installer. The installer’s group checkboxes and group radio buttons will be set to this configuration each time it is launched. They will also be set to this configuration whenever the user clicks on the “Standard” button in the installer’s Main Dialog or AltMain Dialog.

Build Installer

Use the “Build Installer” button to create a complete set of distribution disks. Before you create a disk set, you should prepare a disk to be used as the first disk in the set. First, give the disk an appropriate name (you will NOT be able to change the name later). Next, remove any files which you do not want included in the final distribution—the Builder will not delete any files on this disk. If you want to include other files on this disk (e.g. a “Read Me” file) you should add them now since the Builder will typically fill this disk completely full. Finally, click the “Build Installer” button to start the build process.

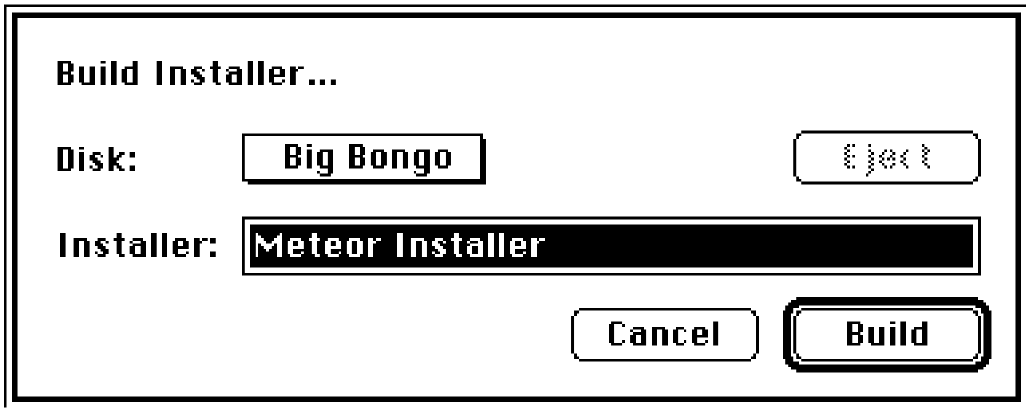


Figure 11: Build Installer Dialog

The Builder will display the dialog in Figure 11. Use the disk pop-up menu to select the first disk of the set. You may specify the name of the installer application in this dialog.

When you click the “Build” button, the Builder will add an installer application to this disk. When the disk is full, the Builder will prompt for additional disks as needed.

WARNING: ALL ADDITIONAL DISKS ARE AUTOMATICALLY ERASED BY THE BUILDER BEFORE USE.

Reducing Build Time. While you are developing your installer, you can reduce the time required to build a test installer by building it on your hard disk rather than floppies.

Updating Disk Sets. If you use the Build Installer command on an existing disk set, the Builder checks the modification date of your source archive. If the archive has not been changed since the last build, the Builder will give you the option of “Updating” the installer on the disk set. During an update, the Builder attempts to save time by replacing only the installer application in your disk set without regenerating all the installer data files.

Since the installer is typically located on a full disk, an update may fail if the new version of your installer is significantly larger than the current version. If this happens, you may have to rebuild the entire disk set.

License

When you order a license for Smaller Installer, you receive a license number for your product. Use the “License” command in the Action menu to enter your license number in your project file, then rebuild your distribution disk set. This will remove the demonstration dialog which is displayed by unlicensed installers.

Tips/Caveats

1) You should rebuild the desktop file on each of your distribution disks before shipping. If you do not rebuild the desktop files, the icons for the installer and installer data files may not be displayed properly. Furthermore, users running System 6 may see alerts stating that the disks are damaged and require minor repair. For best results, you should rebuild the desktop files under System 7.

   To rebuild the desktop file for a floppy disk, insert the disk in the floppy drive while holding the COMMAND and OPTION keys.

2) To ensure each installer data file has been initialized by the Finder, you should open the root folder for each floppy. If the files are not initialized, the view mode may change when loaded on a different system.

3) The installation process will look better if you close the root folder of all disks except the first in a disk set. This will prevent the Finder from zooming open the window on each disk as it is loaded.

4) New applications which are installed in closed folders will not be registered in the Finder's desktop database until the folder containing the application is manually opened by the user. Any document belonging to the application will appear with a generic icon until the application is registered. For best results, you should direct the user to open the application's folder as soon as possible.

5) It is generally a good idea to place all installed files in one folder which the user can easily move to a different destination if desired. If you have help text, it is a good idea to indicate the name of the folder to avoid possible conflicts with any folders the user may already have installed.

6) If possible, give a list of the files that will be installed and summarize any other modifications the installer will make. This helps power users avoid problems and makes everyone more comfortable with the installation process.

7) Adding a ‘vers[1]’ resource to your resource file will allow you to provide a version number and copyright information in the installer's Get Info window.

8) You will typically save space by using object-oriented PICTs rather than bit-mapped PICTs in your resource file; however, be careful not to use unusual

fonts in your PICTs since these will display poorly on systems which do not have the fonts installed.

9) Some virus checking programs will complain when the installer creates or modifies an application. To avoid confusion, it is probably best to add a note to your help text advising users to turn off virus protection programs before starting an installation.

Release History

8-27-93 Version 1.1

- New: Imbedded first data file in installer application.
- New: Added support for external language modules.
- New: Added option to replace files without prompting user.
- New: Added option to ignore creation date of disks.
- New: Added bold outline to OK button in Completion Dialog.
- Change: Removed QuickTest function.
- Bug Fix: Apple menu title was not always refreshed properly after initial launch of installer.
- Bug Fix: Added SetVol in installer code to prevent errors in hook code which uses HOpenResFile (only necessary for multi-disk installs under System 6).

6-28-93 Version 1.0.2

- Bug Fix: Installer would not eject diskettes from an external drive.

1-13-93 Version 1.0.1

- Change: Modified licensing code.

12-21-92 Version 1.0

- First production release.