InstaCompOne 1.1.1
Technical Guide

File and Resource Compression for the Apple Installer

®

Document Version 1.9 - February 12, 1998

Table of Contents

Table of Contents

CHAPTER 1

Overview

This chapter provides an overview of the InstaCompOne suite of tools used for adding compression / decompression support to Installer 4.X scripts.

Overview

The InstaCompOne SDK allows developers writing scripts for Apple's Installer 4.0.3 application to split and compress their source files, fonts and resources, then have them transparently decompressed and joined during the installation.

This document assumes you are familiar with MPW (Macintosh Programmers Workshop) and have experience writing Installer 4.X scripts. See the document "Installer 4.0.3 Technical Guide" for detailed information on writing scripts for Installer 4.0.3.

Features

The InstaCompOne 1.1.1 includes the following features:

- **Competitive compression savings.** Average savings range from 40% to 60%.
- **Decompression is fast and transparent to the user.** Decompression is performed entirely in memory while the data is read from the source file, freeing the user from any wait at the end of the installation. Scriptwriters can choose between 68K-only, PC-only and fat versions of the decompressor code.
- **Fonts and other resources can be easily compressed.** This allows for additional savings with very low scripting overhead. In many cases it only takes an additional reference to the InstaCompOne Atom Extender from a Resource or Font Atom to enable decompression.
- **Compresses multiple files and resources into a single archive.** This saves disk space, while enabling the scriptwriter to reduce the source files to one file per installation disk. Multiple files and resources can also be stored in a single archive without compression.
- **Split files, fonts, and resources before compression.** This allows efficient use of space within the installation disk set, reducing the need for adding additional installation disks simply because a compressed file would not fit completely onto the remaining space on an installation disk. This feature also allows files that are too large to be compressed onto a single installation disk to be included in the installation.

Changes in Version 1.1.1

Version 1.1.1 includes these changes:

- **Additional file header information.** The archive header now contains information about the region code and localization revision level of each file.
- **Archive modification date bug fix.** The modification date of the archive file now remains unchanged if you list the header of the archive or decompress a file.

Example Script

This example describes the process of creating a very simple Installer document that installs and decompresses a SimpleText file (“Example File”) and a family of fonts (Helvetica). All necessary files required to create a working version of this script are available in the folder “InstaCompOneExample” in the InstaCompOne SDK. The process of building a working Installer script requires three mains tasks: compressing source files and fonts using the InstaCompOne MPW tool, creating and compiling the script file using the Rez MPW tool, and running the ScriptCheck MPW tool on the Installer document.

The commands shown in the discussion below are executed automatically by the Build command. These build commands can be seen in the file “Makefile”. Refer to *Macintosh Programmers Workshop 3.0 Reference* for more information about using the MPW Shell application and the Rez tool.

Compressing the source file and fonts

An example SimpleText file (“Example File”) is used in this example. Assuming the current directory is set to the example folder, we compress the file into an InstaCompOne file archive with the command:

```
InstaCompOneTool “:Orignal Sources:Example File” -o ":Tidbits:Installation Data"
```

After the compression has finished, listing the contents of the archive shows the SimpleText application as its only contents.

```
InstaCompOneTool -o ":Tidbits:Installation Data" -l
ID  File Name                      Type   Crtr   DF Org   DF Cmp  Svd  RF Org   RF Cmp  Svd
———————————————————————————————————————————————————————————————————————————————————————————
  1 'Example File'                'ttro' 'ttxt'     85       55   36     3065     1092  65
```

Next, we must compress our font resources. Each of the eight font resources will be compressed separately and stored as individual items within an InstaCompOne resource archive.

We compress the eight font resources with the commands:

```
InstaCompOneTool “:Orignal Sources:Helvetica” -k sfnt=16033 -a part=200 -o ":Tidbits:Installation Data"
InstaCompOneTool “:Orignal Sources:Helvetica” -k sfnt=5336 -a part=201 -o ":Tidbits:Installation Data"
InstaCompOneTool “:Orignal Sources:Helvetica” -k NFNT=24110 -a part=202 -o ":Tidbits:Installation Data"
InstaCompOneTool “:Orignal Sources:Helvetica” -k NFNT=22271 -a part=203 -o ":Tidbits:Installation Data"
InstaCompOneTool “:Orignal Sources:Helvetica” -k NFNT=14739 -a part=204 -o ":Tidbits:Installation Data"
InstaCompOneTool “:Orignal Sources:Helvetica” -k NFNT=15203 -a part=205 -o ":Tidbits:Installation Data"
InstaCompOneTool “:Orignal Sources:Helvetica” -k NFNT=17245 -a part=206 -o ":Tidbits:Installation Data"
InstaCompOneTool “:Orignal Sources:Helvetica” -k NFNT=17892 -a part=207 -o ":Tidbits:Installation Data"
```

After each font resource has been compressed into its resource archive, we can list the contents of several of these archives.

```
InstaCompOneTool -k sfnt=16033 -a part=200 -o ":Tidbits:Installation Data" -l
    Type     ID  Name                            Org Size Cmp Size Svd
──────────────────────────────────────────────────────────────────────
  1 sfnt   16033 Helvetica                          52812    39290  26

InstaCompOneTool Helvetica -k NFNT=24110 -a part=202 -o ":Tidbits:Installation Data" -l
    Type     ID  Name                            Org Size Cmp Size Svd
──────────────────────────────────────────────────────────────────────
  1 NFNT   24110                                     2694     1746  36
```

Finally, we add the 'FOND' resource to the source file. It's best to convert the resource to the type 'iFND', then use the `encodedFONDRsrc` flag in the Font Atom to signal to the Installer that the source 'FOND' has been encoded. We use the following command to encode and copy the 'FOND' resource from the file "Helvetica" into the source file "Installation Data":

```
FONDEncoderTool ":Orignal Sources:Helvetica" -o ":Tidbits:Installation Data"
```

Now we're ready to begin writing the Installer script.

Writing the Installer script

We begin writing our Installer script by including the InstaCompOne Atom Extender resource by adding the line `include "InstaCompOneAtomExt.rsrc" NOT 'vers';` somewhere in our script file. The script resources are created normally with the following important points:

- We must use format 1 versions or higher of the File Atom ('infa') and Font Atom ('inff') resources because these versions allow us to specify the Atom Extender ID number. The number 241 tells Installer 4.0.3 to look for the Atom Extender resource ('inex') with this ID.
- Size, Finder attributes, and resource attribute fields are left as zero. The ScriptCheck MPW tool will fill these fields in later.
- When decompressing font resources, we must provide the Installer with the source resource type and ID. This requires us to specify each strike (size and style) individually. InstaCompOne compressed resources should always use the resource type 'part'. The resource ID (or part number) can be any number between 128 and 32000, and should be a unique part number assigned by the scriptwriter.

The Installer script Rez file "InstaCompOneExample.r" is shown below in its entirety. Please note that, for sake of simplicity in demonstrating compression, this example uses old style (pre-4.0 installer) method for creating Custom Install packages. This example is not intended as an example of how to take advantage of 4.0.3 Installer interface features.

```
#include "InstallerTypes.r"
include "InstaCompOneAtomExt.rsrc" NOT 'vers';

/*************************** Package resources **********************************/
resource 'inpk' (20037) {
    format0 {
        showsOnCustom,    notRemovable, dontForceRestart, 0, 0,
        "Example File Decompression",
        { 'infa', 1001 }
    }
};

resource 'inpk' (20038) {
    format0 {
        showsOnCustom, notRemovable, dontForceRestart, 0, 0,
        "Helvetica Font Decompession Example",
        { 'inff', 1001 }
    }
```

```
};

/************************** Example File Atom *****************************/
resource 'infa' (1001) {
    format1 {
        deleteWhenRemoving, deleteWhenInstalling, copy, dontIgnoreLockedFile,
        dontSetFileLocked, useVersProcToCompare, srcNeedExist, rsrcForkInDataFork,
        leaveAloneIfNewer, updateExisting, copyIfNewOrUpdate, rsrcFork, dataFork,
        0,                /* Total size will be filled in by ScriptCheck */
        0,                /* Finder attributes will be filled in by ScriptCheck */
        10000,
        {   20000, 0, 0 },    /* sizes will be filled in by ScriptCheck */
        0x0,
        0,
        241,
        ""
    }
};

/********************** Example Helvetica Font Atom ***************************/
resource 'inff' (1001) {
    format2 {
        deleteWhenRemoving, deleteWhenInstalling, copy, encodedFONDRsrc, noTgtRequired,
        updateExisting, copyIfNewOrUpdate, dontIgnoreProtection, srcNeedExist,
        byID, nameNeedNotMatch,
        10002,     /* Target */
        20000,     /* Source */
        32,            /* FOND Attributes */
        0,         /* Total font size, filled in by ScriptCheck */
        21,            /* FOND ID */
        explicitFamilyMembers {{
             0, noExtendedStyle, noCondensedStyle, noShadowStyle, noOutlineStyle,
    noUnderlineStyle, noItalicStyle, noBoldStyle, 'sfnt', 32,
            { 20000, 'part', 200, 0, "" },
             0, noExtendedStyle, noCondensedStyle, noShadowStyle, noOutlineStyle,
    noUnderlineStyle, noItalicStyle, boldStyle,    'sfnt', 32,
            { 20000, 'part', 201, 0, "" },
             9, noExtendedStyle, noCondensedStyle, noShadowStyle, noOutlineStyle,
    noUnderlineStyle, noItalicStyle, noBoldStyle, 'NFNT', 32,
            { 20000, 'part', 202, 0, "" },
            10, noExtendedStyle, noCondensedStyle, noShadowStyle, noOutlineStyle,
    noUnderlineStyle, noItalicStyle, noBoldStyle, 'NFNT', 32,
            { 20000, 'part', 203, 0, "" },
            12, noExtendedStyle, noCondensedStyle, noShadowStyle, noOutlineStyle,
    noUnderlineStyle, noItalicStyle, noBoldStyle, 'NFNT', 32,
            { 20000, 'part', 204, 0, "" },
            14, noExtendedStyle, noCondensedStyle, noShadowStyle, noOutlineStyle,
    noUnderlineStyle, noItalicStyle, noBoldStyle, 'NFNT', 32,
            { 20000, 'part', 205, 0, "" },
            18, noExtendedStyle, noCondensedStyle, noShadowStyle, noOutlineStyle,
    noUnderlineStyle, noItalicStyle, noBoldStyle, 'NFNT', 32,
            { 20000, 'part', 206, 0, "" },
            24, noExtendedStyle, noCondensedStyle, noShadowStyle, noOutlineStyle,
    noUnderlineStyle, noItalicStyle, noBoldStyle, 'NFNT', 32,
            { 20000, 'part', 207, 0, "" }
        }},
        241,
        "",
        "Helvetica"
    }
};

/********************* Example File Target File Spec. ***************************/
resource 'intf' (10000) {
    format0 {
        noSearchForFile, typeCrNeedNotMatch, 'ttro', 'ttxt', 0,
        "folder-user:Example File"
    }
};

/******************** Font Suitcase Target File Spec. *************************/
resource 'intf' (10002) {
    format0 {
        noSearchForFile, typeCrNeedNotMatch, 'FFIL', 'DMOV', 0,
        "folder-user:Example Fonts"
    }
};

/************************ Archive Source File Spec. ***************************/
resource 'infs' (20000) {
    'ircp', 'kakc', 0x1, noSearchForFile, TypeCrMustMatch,
    "Tidbits:Installation Data"
};

/************************* Preference Resource ********************************/
resource 'inpr' (300) {
    format0 {
        useFolderTargetMode, dontAllowUserToSetSystemDisk, showSelectedSizeInCustom,
        noSetupFunctionSupplied, dontAllowCleanInstall, dontAllowServerAsTarget,0,0,
            {   301, 311, 301, 311, 302, 312, 302, 312 },
        "Example Folder"
    }
};
```

We compile the Rez code using the Rez MPW tool by executing the following command

```
Rez InstaCompOneExample.r -o InstaCompOneExample -t 'kajr' -c 'kajr'
```

After compiling the Rez code with the Rez MPW tool, we will have an Installer document, but it still needs to have the file and resource information updated.

Finishing with ScriptCheck

The ScriptCheck MPW tool performs two important actions on your script:

- Checks for syntax errors, such as referencing a script resource that is not present in the Installer document. The ScriptCheck tool also verifies that all source files and resources exist on the source disks.
- Fills in the size, Finder attributes, and resource attribute fields we left as zero when we wrote the script. The ScriptCheck MPW tool gets this information by asking the code contained in the ScriptCheck extension file.

To begin the checking process we execute the command:

```
ScriptCheck 'InstaCompOneExample' -h -d -a
```

The ScriptCheck MPW tool is described in detail in the document “ScriptCheck 4.0.3 Guide”.

Using ScriptCheck with InstaCompOne Archives

To assist ScriptCheck in determining file and resource information of items contained within InstaCompOne archives, an extension file has been provided for ScriptCheck. This file enables ScriptCheck to determine size and attribute information about the original item contained within an InstaCompOne archive.

The file “InstaCompOneSCExt.rsrc” contains code that can retrieve the required information from the archive to enable ScriptCheck to correctly update the Installer document. This file is included in the InstaCompOne SDK.

To enable ScriptCheck for use with InstaCompOne archives :

- Place a copy of the file “InstaCompOneSCExt.rsrc” in the same folder as your Installer script source file (myInstallScript.r, etc.).
- Rename the file so that it has the same name as the install script source but uses the filename extension of “.scx”. Example : “myInstallScript.scx”

When running ScriptCheck to build the installer for InstaCompOneExample, ScriptCheck looks for a file named “InstaCompOneExample.scx” which was created in our automated build process.

C H A P T E R 2

InstaCompOne MPW Tool

This chapter describes the InstaCompOne MPW Tool.

About InstaCompOneTool

You'll use the InstaCompOne MPW Tool named "InstaCompOneTool" to compress your files and resources into an archive. The InstaCompOne MPW Tool also allows you to list, remove and decompress files or resources contained in the archive.

The actual compressed data is stored differently depending on whether you are compressing files or resources:

- For files: the compressed data is stored in the data fork of an archive file.
- For resources and fonts: the compressed data is stored in a archive resource usually of type 'part'. These archive resources can be placed in any file, including an archive file that contains compressed file data in its data fork.

You'll use the InstaCompOneTool to compress and manage both archived files and archived resources. The rest of this chapter is divided in two sections: using the InstaCompOneTool with file archives and using the InstaCompOneTool with resource archives.

The InstaCompOne MPW Tool requires MPW 3.0 or newer and System 7.0 or newer. To list the help text, invoke the tool without parameters.

Using the InstaCompOneTool with File Archives

This section describes how to use InstaCompOneTool to compress and manage compressed files stored in a file archive.

Command Line

Usage when compressing files in a file archive:

```
InstaCompOneTool filename -o archiveFilename [-nc] [-e] [-f name]
```

Usage when decompressing, listing or removing files from a file archive:

```
InstaCompOneTool [-d|-l|-ll|-r] filename [-i ID] -o archiveFilename
```

Tool Options for File Archives

`fileName ...`	When compressing, specifies one or more files to be compressed and added to the archive file. When decompressing or removing, specifies one or more file entries in the file archive. If more than one entry exists in the archive with the same filename, you may need to use the -i option to specify the exact entry. The filename can be a single file name, partial path, or full path.
`-o archiveFileName`	Specifies a filename, partial path, or full path to a new or existing file archive.
`-f fileName`	Specifies the filename to use in the archive. This allows a file to be renamed as it is added to an archive. This is handy if your source file has a different name than the target file.
`-i entryID`	Specifies a compressed file entry in the archive using the file's unique identifier, instead of the name. This option can only be used when removing or decompressing a file entry in the archive and can be mixed with filenames or the -k option. You can use multiple -i options in one command line.
`-r`	Requests that the specified compressed file be deleted from the archive indicated by the -o option. The compressed files can either be specified by filename or using the -i option.
`-d`	Requests that the specified compressed file be decompressed to a file from the archive specified in the -o option. The decompressed file is placed at the location specified in the file name path. An existing file with the same name will be replaced without warning.
`-l`	Writes a listing of the files contained in the archive to stdout. The -l option should only be used with the -o option.
`-ll`	Writes an extended listing of the files contained in the archive to stdout. The -ll option should only be used with the -o option.
`-nc`	Adds files to an archive without compression. This allows multiple files to be stored in a single archive file without the overhead of compression or decompression.
`-e`	Requests that the data fork being compressed from the specified source files actually be stored in the file archive as coming from the resource fork. This option is required when compressing a resource fork that has been split using the FileAndRsrcSplitterTool, because this tool always places the split resource data into the data fork of the split files.
`-f newFilename`	Renames the source file being compressed to that specified in newFilename. Since the InstaCompOne Atom Extender finds the file in the file archive based partly on the target file name, this eliminates the need to rename split files using MPW or the Finder before compressing.

Compressing Files

You can add one or more files to a new or existing archive file by specifying the files to be compressed and the archive filename. If the specified archive file does not exist, it will be created. The tool will not create folders that do not exist. If a file exists with the same name as the specified archive name, but the file is not a valid archive, an error is written to stderr. If the folders or directories specified in the path to a non-existing archive do not exist, the folders or directories will not be created, and an error is will be written to stderr.

For example, to compress the files MyBigFile and “A Large File” and add them to the archive file “Compressed Data” execute the command:

```
InstaCompOneTool  MyBigFile "Hard Disk:A Large File" -o "Compressed Data"
```

If an entry for the file you are compressing and adding to the archive already exists, it will be replaced. The tool uses the name, type and creator to determine if the file is the same as one already in the archive. Each time you add or update a file in the archive a new ID is assigned to the compressed file entry.

NOTE

The suggested MPW Shell partition size is at least 4 Mb’s. Compressing large files may require an even larger partition size. u

Viewing a File Archive

You view the contents of a file archive using the -l option.

For example, to list the information about the compressed files in the file archive “Compressed Data” execute the command:

```
InstaCompOneTool   -o "Compressed Data" -l
```

The listing is written to stdout.

```
ID  File Name               Type  Crtr   DF Org   DF Cmp  Svd  RF Org   RF Cmp  Svd
----------------------------------------------------------------------------------
  7 General Controls        cdev  misc        0        0  0    58658    20700  65
  9 Labels                  cdev  flbs        0        0  0     2922      915  69
 12 Mouse                   cdev  mous        0        0  0    18339    15015  19
 20 Views                   cdev  fvew        0        0  0     2921      841  72
----------------------------------------------------------------------------------
            TOTALS                            0        0  0    82840    37471  45
```

Removing Files from a File Archive

Compressed files can easily be removed from an archive file by using the -r option. Specify the files by listing their filenames in the command, or use the -i option to refer to a compressed file using its unique identifier. This may be necessary when several compressed files have the same name, otherwise the first file found in the archive matching the name will be removed.

For example, let’s assume we start with the archive:

```
ID  File Name               Type  Crtr   DF Org   DF Cmp  Svd  RF Org   RF Cmp  Svd
----------------------------------------------------------------------------------
  1 LaserWriter             PRER  LWRW        0        0  0   268960   169589  37
  2 LaserWriter 300         PRER  LWL2        0        0  0   318151   144917  55
  3 LW Select 310           PRES  lwsl        0        0  0   268973   171945  37
  4 Personal LaserWriter SC PRER  LWSC        0        0  0    72716    37332  49
  5 StyleWriter II          PRER  IJR2        0        0  0   301358   137184  55
  6 LaserWriter             TEXT  ????    11311     3793 67      571      336  42
  7 LaserWriter II NT       TEXT  ????    11830     3600 70      571      346  40
  8 LaserWriter II NTX      TEXT  ????    11957     3679 70      571      346  40
  9 LaserWriter Pro 810f    TEXT  ????    29434     6207 79      571      414  28
```

```
 10 LaserWriter Select 360  TEXT  ????    20738     5269  75      573      396  31
──────────────────────────────────────────────────────────────────────────────────
            TOTALS                        85270    22548  74  1233015   662805  47
```

To remove the StyleWriter II and LaserWriter files from the archive we execute the command:

```
InstaCompOneTool  "StyleWriter II" -i 6  -o "Compressed Data" -r
```

Since the name LaserWriter shows twice, we used the option -i 6 to specify the exact entry we wish to delete.

After the remove operation is complete we have the archive:

```
ID  File Name               Type  Crtr   DF Org   DF Cmp  Svd  RF Org   RF Cmp  Svd
──────────────────────────────────────────────────────────────────────────────────
  1 LaserWriter             PRER  LWRW        0        0   0   268960   169589  37
  2 LaserWriter 300         PRER  LWL2        0        0   0   318151   144917  55
  3 LW Select 310           PRES  lwsl        0        0   0   268973   171945  37
  4 Personal LaserWriter SC PRER  LWSC        0        0   0    72716    37332  49
  7 LaserWriter II NT       TEXT  ????    11830     3600  70      571      346  40
  8 LaserWriter II NTX      TEXT  ????    11957     3679  70      571      346  40
  9 LaserWriter Pro 810f    TEXT  ????    29434     6207  79      571      414  28
 10 LaserWriter Select 360  TEXT  ????    20738     5269  75      573      396  31
──────────────────────────────────────────────────────────────────────────────────
            TOTALS                        73959    18755  75   931086   525285  44
```

Decompressing Files

You'll normally use the Atom Extender that you've included in your Installer script to perform the decompression during the installation, but the tool can also be used to decompress any file contained in a file archive.

To decompress a file, use the -d option and specify one or more filenames or file entries using the -i option. Files that are decompressed remain in the archive.

If you do not specify a file to be decompressed, all files will be decompressed. The tool places the decompressed files into the current directory and will replace any file with the same name without notification.

For example, to decompress the files MyBigFile and "A Large File" from the archive file "Compressed Data" execute the command:

```
InstaCompOneTool  MyBigFile "Hard Disk:A Large File" -o "Compressed Data" -d
```

Using the InstaCompOneTool with Resource Archives

This section describes how to use the InstaCompOneTool to compress and manage compressed resource data stored in a resource archive.

Command Line

Usage when compressing and adding or replacing a resource in a resource archive:

```
InstaCompOneTool filename -o archiveFilename [-nc] -k restype=ID -a
restype=ID
```

Usage when decompressing, listing or removing a resource from a resource archive:

```
InstaCompOneTool [-d|-l|-r] filename -o archiveFilename [-i ID] [-k
restype=ID] -a restype=ID
```

Tool Options for Resource Archives

`fileName`	When compressing, specifies the file that contains the resources to be compressed, or when decompressing, the file that the original resources will be placed. The filename can be a single file name, partial path, or full path.
`-k restype=ID`	Specifies the type and ID of a resource to be compressed, decompressed or removed. For example, if the resource has the type 'sfnt' and the ID 5302, then the parameter should be formatted: -k sfnt=5302.
`-o archiveFileName`	Specifies a filename, partial path, or full path to a new or existing file that contains the archive resource.
`-a restype=ID`	Specifies the type and ID of the archive resource.
`-i entryID`	Specifies a compressed resource in the archive using the entry's unique identifier, instead of the resource type and ID. This option can only be used when removing or decompressing a compressed resource entry in the archive resource.
`-r`	Requests that the specified compressed resources be deleted from the archive identified by the -o and -a options. Specify the resources to be removed by using the -k or -i option.
`-d`	Requests that the specified compressed resources be decompressed to a file. The decompressed resources are placed in the filename found in the command. An existing resource with the same type and ID will be replaced without warning.
`-nc`	Adds resources to an archive without compression. This eliminates the overhead of compression or decompression.
`-l`	Writes a listing of the resources contained in the resource archive to stdout.

Compressing Resources

Compressing resources into a resource archive is similar to compressing files, except the archive is actually contained in a resource you choose. You'll need to specify the archive resource type and ID using the -a option. You must also specify the name of the file that contains, or will contain this resource using the -o option.

Multiple resources (except font resources) can be placed into one resource archive, and will be found automatically by the Atom Extender and decompressed during installation. Both the MPW Tool and the Atom Extender use the original type and ID to find the resource data in the archive.

For example, to compress the resource (type 'fred' and ID 128) found in the file "FredRsrcs" and place it into an archive resource of type 'part' and ID 300 in the file "Compressed Fred Data" execute the command:

```
InstaCompOneTool  FredRsrcs -k fred=128 -a part=300 -o "Compressed Fred Data"
```

Each time you add or update a resource in the archive a new ID is assigned to the compressed resource entry.

Viewing a Resource Archive

You can view the compressed resources currently contained in resource archive using the -l option.

To list information about the compressed resource in the archive resource (type ‘part’ and ID 200) in the file “Compressed Data” execute the command:

```
InstaCompOneTool  -a part=200 -o "Compressed Data" -l
```

The listing is written to stdout.

```
    Type     ID  Name                                Org Size Cmp Size Svd
──────────────────────────────────────────────────────────────────────────
  1 sfnt    5336 Helvetica Bold                         51624    38356  26
```

Removing Resources from a Resource Archive

To remove a resource entry from an archive specify these four options in the command:

- -r option, to signify that this is a remove operation
- -o option with the filename of the file containing the resource archive
- -a option with the resource archive type and ID
- -k option with the original, uncompressed resource type and ID to remove

For example, to remove the resource ‘fred’ ID 245 from the resource archive ‘part’ ID 128 in the file “Compressed Resources” execute the command:

```
InstaCompOneTool  "Hard Disk:My Resources" -k fred=245 -a part=128 -o "Compressed Resources" -r
```

Decompressing Resources

To decompress a resource specify these five options in the command:

- -d option, to signify that this is a decompression operation
- -o option with the filename of the file containing the resource archive
- -a option with the resource archive type and ID
- -k option with the original, uncompressed resource type and ID to decompress
- -filename or pathname of the file in which the original resource will be placed

For example, to decompress the resource ‘fred’ ID 245 to the file “My Resources” on the disk “Hard Disk” from the resource archive ‘part’ ID 128 in the file “Compressed Resources” execute the command:

```
InstaCompOneTool  "Hard Disk:My Resources" -k fred=245 -a part=128 -o "Compressed Resources" -d
```

Compressing Font Resources

Compressing font resources is very similar to compressing any other resource, except several limitations must be kept in mind. There are four common font resource types: ‘FOND’, ‘FONT’, ‘NFNT’ and ‘sfnt’. The ‘FOND’ resource describes how the other types relate, and the ‘FONT’ is an older type that is rarely used today. Therefore, the most common font resources you will compress will either be of type ‘NFNT’ or ‘sfnt’.

NOTE

Keep these important notes in mind when compressing font resources:

- Multiple font resources cannot be placed into a single resource archive ; therefore, one resource archive must be created for each original font resource. Although, any number of resource archives can be placed into a single source file, even into a file that contains a file archive in its data fork
- The ‘FOND’ resource should never be compressed or split. It’s also best to encode the ‘FOND’ resource as type ‘iFND’ whenever it’s ‘NFNT’, ‘sfnt’ or ‘FONT’ resource are compressed. An example of this encoding is described below. u

For example, to compress the resource (type ‘sfnt’ and ID 5336) found in the file “Helvetica” and place it into an archive resource of type ‘part’ and ID 200 in the file “Compressed Data” execute the command:

```
InstaCompOneTool  Helvetica -k sfnt=5336 -a part=200 -o "Compressed Data"
```

The Installer will be expect to the ‘FOND’ resource located in the file you have specified in your ‘inff’ script resource. You can use the FONDEncoderTool or Rez commands to encode and copy the ‘FOND’ resources. To encode the ‘FOND’ resource and store it in the file “Compressed Data” execute the command:

```
FONDEncoderTool Helvetica -o "Compressed Data"
```

or

```
echo "include Helvetica 'FOND' AS 'iFND';" | Rez -o "Compressed Data" -append
```

The ‘FOND’ resource will be stored as a resource of type ‘iFND’ with the identical resource ID as the original ‘FOND’ resource.

NOTE
The ‘FOND’ encoding is only a feature of format version 2 of the Font Atom. Make sure to use the `encodedFONDRsrc` flag to signal to the Installer that the source ‘FOND’ has been encoded. u

C H A P T E R   3

InstaCompOne Atom Extender

This chapter describes the InstaCompOne Atom Extender.

Using the InstaCompOne Atom Extender

The InstaCompOne Atom Extender is two resources you add to your Installer script file that will be called at the appropriate time to decompress (and optionally to join split pieces of) a File, Resource or Font Atom’s data during the installation. This Atom Extender can only be used with archives created using the InstaCompOne MPW Tool.

Deciding which InstaCompOne Atom Extender to use

Three flavors of the InstaCompOne Atom Extender are available trade off decompression speed versus Installer script size. You should choose the one the best suits your needs:

- 68K-only ('inex' ID 241): Best for scriptwriters that want to upgrade from the old 68K decompressor with minimal effort, or don’t have disk space to add the fat version to the Installer script file.
- PPC-only ('inex' ID 242): Best for scriptwriters that will only install on PowerPC machines and want maximum speed with minimum-sized Installer script.
- “Fat” ('inex' ID 243): Best for the majority of scriptwriters. The user benefits from additional speed on PPC machines at the expense of additional code in the Installer script.

You can actually use all three in the same script, but scriptwriters will normally use only one.

Adding the InstaCompOne Atom Extender to your Installer Script

The InstaCompOne Atom Extender resources are contained in the file “InstaCompOneAtomExt.rsrc”. Once you have determined which flavor of the InstaCompOne Atom Extender you want to use, you’ll need to place these resources into your Installer script file so the Installer can find it.

For example, to include the "Fat" flavor of the InstaCompOne Atom Extender in your Installer script Rez file (".r" file), include the following lines:

```
include "InstaCompOneAtomExt.rsrc" 'inex' (243);
include "InstaCompOneAtomExt.rsrc" 'exfn' (243);
```

For those File, Resource and Font Atoms that are copying data contained in an InstaCompOne archive, place the appropriate number (241, 242, or 243) in the Atom Extender ID field of the atom resource. This field is only available in format 1 or higher versions of the 'infa' , 'inra' and 'inff' script resources.

NOTE

For File Atoms, you'll need to use the `rsrcForkInDataFork` flag to enable the Atom Extender to access the archive correctly when copying the resource fork of a compressed file. u

How the Atom Extender Locates Data Inside the Archive

The InstaCompOne Atom Extender uses different criteria to locate a file or resource in the archive than does the InstaCompOne MPW tool.

The Atom Extender uses the following strategy for find the correct data in the archive:

- For files, the Atom Extender uses only the filename and the original, uncompressed size of the data. The filename will be extracted from the target path in the 'intf' target spec resource for the file atom ('infa').
- For font resources, only the target type of the resource is used. This limitation prevents multiple compressed font resources from being stored in a single resource archive.
- For non-font resources, both the target type and target ID are used to find the entry inside the resource archive.

Most of the time you'll never have to worry about how the data is found in the archive, because it just works!

NOTE

You may notice some degradation of decompression speed from floppy disks if a single archive contains more than 30 files. This will be addressed in a future version of the InstaCompOne Atom Extender. u

Debugging Decompression Problems

If something goes wrong during decompression, the Installer will cancel the installation with a generic error message. During your testing cycle, it's easiest to determine the actual problem using the Installer Debugger supplied with the Installer 4.0.3 SDK. The InstaCompOne Atom Extender will write any error information to the Installer Debugger main window.

InstaCompOne Atom Extender error codes:

Error# 28201	The file archive or resource archive may be corrupted or is not a valid archive. The Atom Extender looks for a signature in the header of the archive, and if not found returns this error. If the signature is correct but the checksum of the header and catalog is incorrect the archive is assumed corrupted and this error is generated.
Error# 28202	The compressed file or resource was not found in the catalog of the archive. This is most often because the compressed file's

	filename is incorrect in the archive when using with a File Atom. See the above section “How the Atom Extender Locates Data Inside the Archive” for information about our catalog lookup strategy.
Error# 28203	The checksum of the data that has been read does not match the value stored in the archive. The archive may be corrupted.
Error# 28204	This version of the InstaCompOne Atom Extender does not support the archive version. This might be the case if you are using an older Atom Extender with an archive created with a newer version of the compressor tool.

CHAPTER 4

InstaCompOne ScriptCheck Extension

This chapter describes the InstaCompOne ScriptCheck Extension.

In order to have ScriptCheck compute the correct target size values and fill in the appropriate Finder flags and version number for a compressed file, ScriptCheck must know how to extract this information from the archive file.

Using the InstaCompOne ScriptCheck Extension

To enable ScriptCheck for use with InstaCompOne archives :

- Place a copy of the file "InstaCompOneSCExt.rsrc" in the same folder as your installer script source (myInstallScript.r, etc.).
- Rename the file so that it has the same name as the install script source but uses the filename extension of ".scx". Example : "myInstallScript.scx"

NOTE

You must use version 4.0.1 or newer of the ScriptCheck tool when using the InstaCompOne ScriptCheck Extension. u

When checking a File, Resource or Font Atom which references an Atom Extender, ScriptCheck looks for a file named "yourScriptFileName.scx", where *yourScriptFileName* is the same as the Installer Script file name you are checking. To create the ScriptCheck Extension for the InstaCompOne compressor, just rename the file "InstaCompOneSCExt.rsrc" to "yourScriptFileName.scx". Place this file in the same directory as the Installer script you are checking.

See the document "ScriptCheck 4.0.3 Guide" for the gory details of how ScriptCheck extensions work.

See the section "InstaCompOne Atom Extender" in this document for a listing of the possible error code returned from the InstaCompOne ScriptCheck extension.

C H A P T E R 5

FileAndRsrcSplitterTool

This chapter describes the FileAndRsrcSplitterTool.

About the FileAndRsrcSplitterTool

You'll use the MPW Tool named "FileAndRsrcSplitterTool" to split large files and resources to enable floppy disk sets to be created, or to optimize the disk space usage. Additional disk space savings are possible when the split pieces are compressed using the InstaCompOne compression tools.

The FileAndRsrcSplitterTool can split either files or individual resources. The rest of the this chapter is divided between it use with files and its use with resources.

NOTE

The FileAndRsrcSplitter MPW Tool requires MPW 3.0 or newer and System 7.0 or newer. To list the help text, invoke the tool without parameters. u

Using the FileAndRsrcSplitterTool with Files

This section describes how to split files for use with Installer 4.X. File Atom format version 1 or higher supports multiple spilt sources. Refer to the Installer 4.X Technical Guide for additional information concerning installation of split files.

Command Line

```
FileAndRsrcSplitterTool filename… -s sizeInBytes [-s# partSizeInBytes]
[-o outputFilename]
```

Tool Options for Splitting Files

`fileName ...`	Specifies one or more files to be split. The filename can be a single file name, partial path, or full path. **NOTE** If specifying multiple source files to be split, do not use the -o

option. This will allow the source filename to be used as the root name of the split files. u

`-s sizeInBytes`	Specifies in bytes the maximum size desired for each resulting piece of the file or files to be split.
`-s# sizeInBytes`	Specifies the maximum part size in bytes for a specific part, where '#' is a number from 1 to 999. For example, to create three parts of sizes 100, 200, and 300 bytes, use the options: -s1 100 -s2 200 -s3 300.
`-o outputFilename`	Specifies the root name and the location of the created split files. The filename can be a single file name, partial path, or full path.

NOTE
If specifying multiple source files to be split, do not use the -o option. This will allow the source filename to be used as the root name of the split files. u

The split file names are created by appending the appropriate suffix to the root filename along with the split number. If a resource fork is present in the source file, split files will be created with the suffix ".rsrc1", ".rsrc2", etc. If a data fork is present in the source file, split files will be created with the suffix ".data1", ".data2", etc. For example, if the specified output filename was "splitText" and the original file was split into four pieces, the filenames of the resulting files might be "splitText.rsrc1", "splitText.rsrc2", "splitText.rsrc3" and "splitText.data1".

Splitting files into equal sizes

You can split one or more files into separate pieces by specifying the files to be split, the maximum size in bytes for each split piece, and an optional object filename. If an object filename is specified, resulting files containing the split pieces of the original file will have '.rsrc1', '.rsrc2', etc. appended to the specified object filename. If no object filename is specified, resulting files will use the filename of the original file with '.rsrc1', '.rsrc2', etc. appended.

For example, to split the file "TeachText" into one or more pieces that are all smaller than 32,000 bytes execute the command:

```
FileAndRsrcSplitterTool  TeachText -s 32000 -o "splitTeachText"
```

This will generate two files named "splitTeachText.rsrc1" and "splitTeachText.rsrc2". If a file exists with the same name as a file generated by file splitting , it will be replaced by the generated file without warning.

To instruct the Installer to join your split files into one target files during the installation you'll need to create an entry in the File Atom's source list for each split file. If the original source file had both a data fork and a resource fork you'll need to create two File Atoms, one to copy the resource fork and another one to copy the data fork. This is necessary to allow ScriptCheck to correctly update the sizes. Since the split resource fork will actually be stored in the data fork of the split file you'll need to use the `rsrcForkInDataFork` flag in your File Atom.

NOTE

When splitting multiple files it is necessary to use the default object filename to avoid having each set of split files being given the same set of filenames. This is accomplished by omitting the -o option (object filename). u

Splitting files into custom sizes

If you need to split your file into unequal sizes, specify the size of each piece using the -s# option.

For example, to split the file "TeachText" into one or more pieces with the first piece being 10,000 bytes in size, execute the command:

```
FileAndRsrcSplitterTool  TeachText -s 32000 -s1 10000 -o "splitTeachText"
```

The file "splitTeachText.rsrc1" will be 10,000 bytes in size, and all other files will be no more than 32,000 bytes in size.

Compressing Split Files

Split files can optionally be compressed using the InstaCompOneTool then decompressed and joined automatically during installation.

The InstaCompOne decompression Atom Extender treats compressed split file just like any other file, but two important points will make sure your file archive is setup properly.

- The name of the file in the file archive must be the same as the target file name as specified in the target file spec. Since the FileAndRsrcSplitterTool creates split files with unique names, the -f option will allow you to easily compress each piece and stored with the target name. Because InstaCompOne archives allow only one item in each archive with the same filename, type and creator, it will also be necessary to compress each split piece of a file into separate file archives.
- The InstaCompOneTool maintains information about which compressed data is from the resource fork and which is from the data fork. Since all split file data is stored in the data fork of the split files you are compressing, you'll need to tell the InstaCompOneTool which is actually from the resource fork.

For example, to compress the split files "splitTeachText.rsrc1" and "splitTeachText.rsrc2" and add them to two archive files "Compressed Data1" and "Compressed Data2" execute the commands:

```
InstaCompOneTool  splitTeachText.rsrc1  -o ":disk 1:Compressed Data1" -f "TeachText" -e
InstaCompOneTool  splitTeachText.rsrc2  -o ":disk 2:Compressed Data2" -f "TeachText" -e
```

If an entry for the file you are compressing and adding to the archive already exists, it will be replaced. Be careful when compressing split pieces that you specify the intended archive correctly. Adding two split pieces of the same file to an archive will result in one of the split pieces being overwritten by the other split piece.

Using the FileAndRsrcSplitterTool with Resources

This section describes how to split resources for use with Installer 4.X. Resource Atom or Font Atom format version 1 or higher supports multiple spilt sources. Refer to the Installer 4.X Technical Guide for additional information concerning installation of split resource or fonts.

Command Line

```
FileAndRsrcSplitterTool filename -k restype=ID -s sizeInBytes [-s#
partSizeInBytes] -a restype=ID -o outputFilename
```

Tool Options for Splitting Resources

fileName	Specifies the file containing the resource to be split. The filename can be a single file name, partial path, or full path.
-k restype=ID	Specifies the resource type and ID within the specified source file to be split.
-s sizeInBytes	Specifies in bytes the maximum size desired for each resulting piece of the resource to be split.
-s# sizeInBytes	Specifies the maximum part size in bytes for a specific part, where '#' is a number from 1 to 999. For example, to create three parts of sizes 100, 200, and 300 bytes, use the options: -s1 100 -s2 200 -s3 300.
-a restype=ID	Specifies the resource type and initial resource ID to be assigned to resource items created within the specified output file. You may use any unique number between 128 and 32000 as the base ID for the resulting split resources. For example, if the target type and ID given was [-a 'curs'=128] and the resource was split into three pieces, the resulting resource items placed in the specified output file would be : 'curs' (128), 'curs'(129), and 'curs'(130). If you won't be compressing the split resource pieces, we suggest you use the resource type 'part'. If you will be compressing the piece using the InstaCompOneTool, then leave the type the same as the original resource.
-o outputFilename	Specify the filename of the file that will contain the split resource pieces. The filename can be a single file name, partial path, or full path.

Splitting Resources

You can split a resource into separate pieces by specifying the file containing the resource item to be split, the resource type and ID of the resource to be split, the maximum size in bytes for each split piece, the object filename that will contain each split resource piece, and the resource type and base ID for each of the resulting split pieces.

For example, to split the resource of type 'STR#' ID 129 within the file "TeachText", into one or more pieces that are all smaller than 2k bytes, execute the command:

```
FileAndRsrcSplitterTool  TeachText -k 'STR#'=129 -s 2000 -a part=1000 -o "splitResources"
```

This will generate two resources of type 'part' with IDs of 1000 and 1001 in the file named "splitResources". If a resource already exists within the specified object file with the same resource type and ID, it will be replaced without warning.

To instruct the Installer to join your split resources into one target resource during the installation you'll need to create an entry in the Resource or Font Atom's source list for each split resource piece.

Compressing Split Resources

Split resources can optionally be compressed using the InstaCompOneTool then decompressed and joined automatically during installation.

One important point to remember when compressing split resource pieces is that the type and ID of the resource that is compressed must be the same as the target resource type and ID specified in the Resource Atom. Although there are several ways to accomplish this task in your script building process, our examples uses less than obvious MPW trickery to change the type and ID of the split resources before compressing them.

For example, to compress the split resources created within the file "splitResources" (see previous section) and add them to two archive files "Compressed Data1" and "Compressed Data2" execute the commands:

```
echo "include ∂"splitResources∂" 'part'(1000) AS 'STR#'(9100);" | Rez -o tempFile1
InstaCompOneTool tempFile1 -k 'STR#'=9100 -a part=1000 -o "Compressed Data1"
echo "include ∂"splitResources∂" 'part'(1001) AS 'STR#'(9100);" | Rez -o tempFile2
InstaCompOneTool tempFile2 -k 'STR#'=9100 -a part=1001 -o "Compressed Data2"
```

Joining Split and Compressed Files and Resources

It is not possible using the InstaCompOneTool to decompress *and* join files or resources that have been split and compressed. Only the Installer can join split files and resources.

Splitting Strategies

File and resource splitting is handy for reducing the disk count in install disk sets, and necessary for including files and resources that are still too large after compression to fit onto a single install disk. Since files must be split before being compressed you may need to use trial and error to determine the right split size to obtain compressed files the optimize the disk space on your floppies.

Use the following steps to help you split and compress your files to optimize disk space usage:

- First, compress the entire file using the InstaCompOne compression tool. Then calculate the "best guess" of how to split the file using the following formula:

  MaxSplitPieceSize = DiskSize * (OriginalFileSize / CompressedFileSize)

- Run the FileAndRsrcSplitter MPW Tool on the large file, using the MaxSplitPieceSize calculated above. Then compress each split piece using the InstaCompOne MPW Tool.
- If any of the split and compressed pieces are still too large to fit onto the install disks, then, either repeat the splitting process using a smaller value for the maximum size of each split file piece, or customize the size of each piece using the -s# option.

We hope to eliminate this tedious process in a future version of the InstaCompOne tools.